A Conceptual Specimen Architecture for an Intelligent Archive in a Knowledge Building System

Abstract

For the past two years, NASA's *Intelligent Systems Project*, a part of the *Computing, Information and Communications Technology Program*, has sponsored research in a number of technologies broadly related to the theme of *Intelligent Data Understanding* (IDU). One of these research projects concerns Intelligent Archives, with an emphasis on the role of an Intelligent Archive in the context of a Knowledge Building Systems (IA-KBS). After defining a conceptual IA-KBS architecture, the IA-KBS project focused on the potential for the algorithms and technologies being developed by other IDU researchers to realize some of the envisioned IA-KBS capabilities. To that end, it has been proposed that an IA test bed be developed which would clearly demonstrate the applicability and utility of the various research efforts sponsored by the IDU area of the IS project. Previous work surveyed the various IDU research projects and identified opportunities for their associated algorithms, technologies and architectures to contribute to such a test bed. The most important conclusion of that paper was that, depending on the selected scenario*, any one of the research projects reviewed could usefully be included in an appropriately architected test bed environment*.

The current paper carries this work forward by proposing a *Conceptual Specimen Architecture* for the IA/KBS. The paper has four main sections and a brief conclusion. The first section focuses on required functionality, and shows how the IDU research and algorithms logically and functionally fit within the CSA. The second section reviews the current state-of-the-art in cyberinfrastructure technologies, concluding that the necessary components are available to provide the necessary system-level interfaces. The third section proposes three compelling use cases, illustrating by specific example how the IDU research, implemented within an Intelligent Archive, can support KBS objectives. The fourth section represents a transition to the next phase of research by describing an achievable software architecture for the IA-KBS within which the capabilities described in the first three sections could be implemented. The paper concludes that a solid technical basis exists to support prototyping and demonstration of the key IA/KBS technologies.

Table of Contents

Abstract (page 1)

Table of Contents (page 2)

1. Functional Decomposition of the KBS/IA (page 4)
    - 1.1 System Control and Administration
    - 1.2 Import, Export, and Web Interface
    - 1.3 Data Mining
    - 1.4. Event Detection
    - 1.5 Feed-back
    - 1.6 Data Quality
    - 1.7 Virtual Products
    - 1.8 Data Access
    - 1.9 Metadata and Libraries
    - 1.10. Internal Users
    - 1.11 External Users
    - 1.12 IS/IDU Research Projects and Algorithms Applicability to the KBS/IA

2. Emerging Cyberinfrastructure (page 19)
    - 2.1 Introduction
    - 2.2 Service Oriented Architecture and distributed services
    - 2.3 SOA and the Web
    - 2.4 SOA and the Grid
        - 2.4.1 Introduction
        - 2.4.2 Lessons learned
    - 2.5 SOA and interoperable standards
    - 2.6 SOA and OGC technologies
        - 2.6.1 Introduction
        - 2.6.2 Lessons learned

3. Use Cases (page 25)
    - 3.1 Use Case 1: Feed-back
    - 3.2 Use Case 2: Background data mining using **XFM**
    - 3.3 Use Case 3: Large scale data mining for science research
        - *Development, Validation, Registration*
        - *Optimization*
        - *Production*
        - *Considerations*
    - 3.4 Use cases involving remote exploration

4. Top-Level Software Architecture and Specimen Design (page 34)
- 4.1 Block Diagram of the Top-Level Software Architecture
  - 4.1.1 The architecture of traditional archive with no intelligent data understanding
  - 4.1.2 The architecture of standalone archives with limited intelligent data understanding
  - 4.1.3 The architecture multiple connected archives
  - 4.1.4 The service oriented software architecture of knowledge building system and intelligent archive
  - 4.1.5 Top-Level Software Architecture of KBS/IA and Specimen Design
- 4.2 Key Components and Interfaces
  - 4.2.1 Client/user interface layer
  - 4.2.2 IDU algorithm and service layer
  - 4.2.3 Catalog
  - 4.2.4 Archive Storage
- 4.3 Service and Data Flows
- 4.4 OWL-S Examples for Kumar’s Algorithms
  - 4.4.1 Top level service definition
  - 4.4.2 Service profile

5. Conclusions (page 48)

References (page 49)

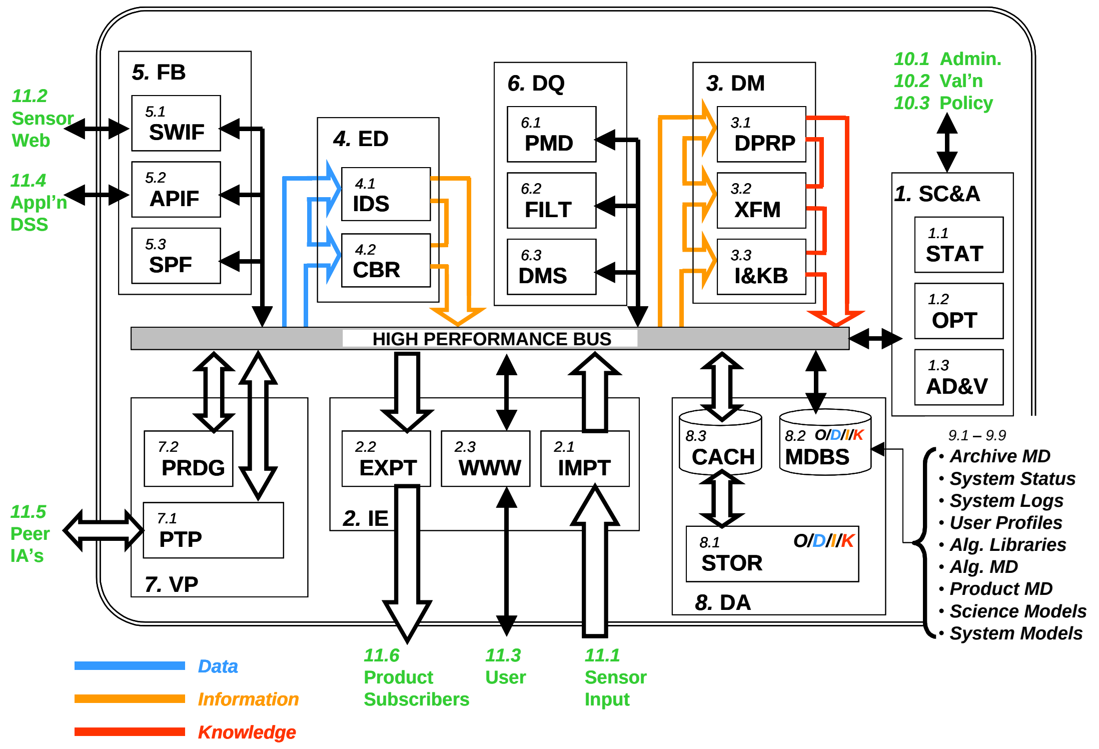
1. Functional Decomposition of the KBS/IA

The purpose of this section is to present a top-level Functional Block Diagram of a Conceptual Specimen Architecture for an Intelligent Archive operating as or within a larger Knowledge Building System.

The discussion will be divided into 12 parts. We begin, in this introduction, with the top-level Functional Block Diagram (FDB), showing the major components (of which there are eight) as well as types of internal and external users and key data structures. In the final section, we will review the Intelligent Data Understanding research projects and associated algorithms examined in previous work [9], and show how these algorithms map onto the FBD – that is, which components and sub-components they might be able to support.

Below is shown the full Top Level Functional Block Diagram for the KBS/IA.

Top-Level Functional Block Diagram of KBS-IA



The numbering scheme within the FBD corresponds to the following sections, in which each of the eight top level functional components, and sub-components, will be described and discussed (§1.1 – §1.8), along with data structures (§1.9), users (§1.10, §1.11), and applicable IS/IDU research programs and algorithms (§1.12)

Because this is a top-level functional diagram, the explicit dependencies as well as data movement and the passing of control are not easily represented. These matters will be touched upon to some extent in the discussion that follows; and to assist the reader, we have provided a diagram of Functional Dependencies (see Table below) between the components and sub-components that may help to clarify the underlying relationships. These will also become clearer when we discuss, in detail, example Use Cases in Section 3 below.

	SC&A			IE			DM			ED		FB			DQ			VP		DA		
	STAT	OPT	AD&V	IMPT	EXPT	WWW	DPRP	XFM	I&KB	IDS	CBR	SWIF	APDS	SP	PMD	FILT	DMS	PTP	PG	STOR	MDBS	CACH
1. SC&A																						
1.1 STAT			X						X													
1.2 OPT	X														X		X	X				X
1.3 AD&V						X	X		X		X							X	X	X	X	
2. IE																						
2.1 IMPT																				X	X	X
2.2 EXPT																				X	X	X
2.3 WWW	X		X						X		X								X		X	
3. DM																						
3.1 DPRP							X	X			X			X		X	X			X	X	
3.2 XFM									X		X				X	X		X		X	X	
3.3 I&KB		X	X							X					X			X	X	X	X	X
4. ED																						
4.1 IDS			X	X			X	X	X			X	X	X	X	X			X	X	X	X
4.2 CBR					X	X	X		X							X			X	X	X	
5. FB																						
5.1 SWIF	X	X		X						X			X		X	X		X	X	X	X	
5.2 APDS				X						X		X			X	X			X		X	
5.3 SP	X									X						X						
6. DQ																						
6.1 PMD		X	X					X										X	X			
6.2 FILT				X	X		X		X	X		X	X	X					X		X	X
6.3 DMS			X																			
7. VP																						
7.1 PTP		X					X		X			X	X		X				X		X	
7.2 PG		X			X	X	X		X		X	X			X			X			X	X
8. DA																						
8.1 STOR			X	X	X		X	X	X	X	X							X	X			X
8.2 MDBS	X	X	X	X	X	X	X	X	X	X	X	X	X		X			X	X	X		X
8.3 CACH		X		X	X	X			X	X	X								X	X		

Functional Dependencies

1.1 System Control and Administration **[SC&A]**

This function manages and optimizes the work of the other functional components. It keeps track of the state of the components, the work queues, and the hardware resources that have been assigned to support the components. It also has models (developed using Data Mining on usage logs) for stochastically estimating future loading and developing optimized work plans. This also comprises the tasks of developing, validating and registering the various knowledge building algorithms that other functional components rely on to perform their tasks.

1.1.1 Current and Estimated System Status **[STAT]**

This function keeps track of the current state of the system: what jobs are running, what hardware resources are utilized and available, what current networking loading is, etc. It also has the capability to predict future loading using KBS models developed from data mining system usage logs. It accomplishes this by using the services of **I&KB** algorithms within the **DM** functional component.

1.1.2 System Optimization **[OPT]**

This function is responsible for automatically optimizing system performance in accordance with current and predicted loading and prioritization models provided by administrative personnel. We might imagine, for example, that this function uses Model Predictive Control (MPC) or related technologies to respond dynamically to uncertain events. This function also provides optimization services to **PTP** to support distributed product generation and **DM** production.

1.1.3 Algorithm Development and Validation **[AD&V]**

This is where algorithms to be used by the various knowledge building components (**DM**, **DQ**, **ED**, etc.) are developed, validated, and registered. This is a manually intensive process, and will typically require interactive monitoring and supervision by scientists and algorithm specialists. This function can build training and test sets by utilizing other capabilities (e.g., **CBR**) in order to construct and optimize KBS algorithm performance. This is also where the metadata describing the performance characteristics of each algorithm is constructed and entered (see §1.9.6).

1.1.4 Other functionality not shown

SC&A is where the user interfaces to monitor the system reside, and where parameters reflecting policy and priorities are entered and maintained. One might imagine log data on cost, reliability, hardware and software configurations, installation and facilities diagrams, etc. All of this standard system management and administration data and software resides in **SC&A**.

1.2 Import, Export, and Web Interface **[IE]**

This is a generic function required by any archive, intelligent or otherwise.

1.2.1 Import **[IMPT]**

This is the interface to the high performance input data stream from any sensor inputs to the IA.

1.2.2 Export **[EXPT]**

This interface to the high performance output network supplies large volumes of data to product subscribers and users and to their systems.

1.2.3 Internet Interface **[WWW]**

This is where the web pages for the IA reside. This includes the development and update of those pages, firewalls, and system security (user registration, authentication, access privileges, cost accounting, etc.).

1.3 Data Mining **[DM]**

This functional area is concerned with the application of information and knowledge building algorithms to large amounts of data resident in IA storage. As such, it must support high volume data flows from and to **DA** resources. The FDB shows a notional flow of ***information*** entering the **DM** functions, and ***knowledge*** flowing out. It is also possible for **DM** to support flows that accept ***data*** (or even ***observations***) as input, and that produce ***information*** (or even ***data***) as output. Thus, **DM** is seen to be a key functional component within the KBS conceptual architecture.

1.3.1 Data Preparation **[DPRP]**

Many data mining algorithms require a variety of reduction and transformation steps before the "raw" inputs, as taken directly from archive storage, are ready to be input to the KB algorithms. This function may therefore utilize the services of a variety of other components: data quality filtering, product generation (and distributed product generation via **PTP**), content based retrieval, etc. The output of this function is typically not permanently stored, but is directly consumed by the **DM** algorithm that requested its services.

1.3.2 Data Transforms **[XFM]**

This function is where non-application-specific algorithms reside. These algorithms transform the input data into a representation in some other basis which can then serve as "raw" input to other **DM** processes. The outputs of **XFM** algorithms will often be permanently stored, and can be considered as a kind of secondary, or shadow, archive sitting beside the data as originally received from the input sensors. Running **XFM** algorithms on data can (and often will) be done speculatively, betting that some as-yet-unknown **DM** process (or user) will be able to find a use for the data as represented in this form. Thus, in one mode of operation, **XFM** algorithms operate more or less continuously, and somewhat blindly, as background processes. As will be seen below (§1.12.2, §1.12.11), IS/IDU research by both Tilton and Emerson fits well into this functional area. **XFM** may also be used by **VP** to generate "product on demand," or by **DPRP** to prepare data (or information) for subsequent processing.

1.3.3 Information and Knowledge Building Algorithms **[I&KB]**

This is the "beating heart" of the KBS functionality – the place at which the various types of data mining and knowledge building algorithms are applied.

Many of the services provided by other components are expressly included so as to enable this component. Algorithms are developed and validated (**AD&V**), scientific priorities are established, and an optimized (and dynamically re-optimized) work flow is created and implemented (**OPT**). Data is then prepared for input (**DPRP**), perhaps drawing on distributed resources (**PTP**), and implementing distributed versions of the **DM** algorithm (again, **PTP**). The results of this process are then made directly available to users (**EXPT** or **APIF**), and may also become part of the permanent archive (**DA**).

It is worth emphasizing a point glossed over above: **I&KB** is where distributed data mining algorithms "reside," but **PTP** is where the negotiations with other IA peers takes place, and **OPT** is where the optimized decisions are made about where, within the GRID, processing should take place. In other words, efficient distributed data mining requires the close co-operation of **I&KB**, **PTP**, and **OPT**, as well of their counter-parts in other peer IA systems.

1.4. Event Detection **[ED]**

As shown in the FBD, notionally we expect this functional component to accept ***data*** as input and produce ***information*** as output. Thus, in the $O \rightarrow D \rightarrow I \rightarrow K$ KBS hierarchy, **ED** will generally operate at a level below that of **DM**. As such, **ED** focuses not so much on finding relationships as on extracting or identifying inherent characteristics or features via pattern matching. Thus, the "event" which is detected is the presence of some feature within the data set being examined. **ED** functions are capable of being applied to very large aggregations of data (for example, the real-time input data stream from the sensor sources) looking for such occurrences. Key performance measures for these types of functions are (1) throughput and (2) Type I and Type II error rates (that is, false positives and false negatives). As with other KBS functions, the algorithms used by **ED** will first undergo a development, validation, and registration process (**AD&V**), and will be invoked in an optimized fashion by **SC&A** control processes.

1.4.1 Input Data Stream **[IDS]**

This functional sub-component is integrally tied into the ingest process. It thus provides a key service to near-real-time feed-back (**FB**) by detecting and reporting occurrences of interest (say) within a larger Sensor Web. It also supports **DQ/FILT** by searching for patterns that may indicate bias or error.

1.4.2 Content Based Retrieval **[CBR]**

Here, the "event" of interest is some screening criteria to be used to down-select from a large data source a subset (presumably small) that satisfies a condition of interest. Note that, in OODB terms, this function might be performed in a massively parallel way via methods stored locally within the data structure itself. Alternatively, the test may be located externally to the data, which passes under the scrutiny of the filter (much as in **IDS**). This function may also be instantiated as an ongoing back-ground process routinely generating metadata on large collections of data for possible as-yet-unspecified future use.

1.5 Feed-back **[FB]**

This functional component is the means by which the KBS/IA participates in near-real-time processes that are largely external to it. Event detection on the input data stream (**IDS**) is an important enabler; but **FB** also provides access, via **PRDG** and **PTP**, to the larger set of distributed archive resources and products.

1.5.1 Interface to the Sensor Web **[SWIF]**

The idea of the Sensor Web is for heterogeneous sensing systems to dynamically co-operate in response to rapidly emerging collection opportunities. A key aspect of this process is the detection of events within one sensor stream that could trigger re-tasking by that sensor or other co-operating sensors. The services to support such functionality are provided to the Sensor Web via this component.

1.5.2 Interface to Application Partner Decision Support Systems **[APIF]**

Sensor systems can play a significant role as inputs to NASA Application Partners (such as FEMA, or the Weather Service). Typically, the interface will be to automated Decision Support Systems maintained by those partners. This functional component not only provides data and alerts to those systems; it also provides access to the services of the IA (including its distributed peers via **PTP**) as well as to the Sensor Web (via **SWIF**).

1.5.3 Sensor Performance Feed-back **[SPF]**

This interface provides feed-back to the input sensor regarding possible bias or other degradation of input product quality. As such, it makes extensive us of **DQ** and **IDS** services.

1.6 Data Quality **[DQ]**

This functional component provides assurance to users of archived data and products that the delivered results are compatible, in pedigree and quality, with the expectations at the user, application, or system interface.

1.6.1 Product MetaData **[PMD]**

When a data product is to be used as input to a KBS algorithm (or other model or process), the scientific integrity of the result may depend upon the precise nature of the "production pedigree" of that input data. For example, models change over time as new features or modeling techniques are incorporated. Hence, a product produced at one point in time using one version of a model may have different numerical or other characteristics from a similar (or similarly named) product generated at some other (earlier or later) time. The purpose of the **PMD** functional component is to maintain the descriptive data structures (see §1.9.7) that capture this pedigree, and to enforce rules reflecting scientific or algorithmic dependencies and constraints involving these features.

1.6.2 Apply Data Quality Filters **[FILT]**

This is a special kind of event detection, where the "event" in question is the presence of some computable feature indicating the presence of a fault or discrepancy. Hence, **FILT** will typically rely on **ED** services. However, the input data need not be (for example) the input data stream (**IDS**), but could be any (perhaps very large) data set extracted from the archive.

1.6.3 Data Mining Support Services **[DMS]**

This is a special use of Data Quality functionality in which (for example) rule based induction algorithms are made more efficient by retraining them on sets where the data has been adjusted to remove outliers. This process (see §1.12.10 below) can be applied to any of a large variety of **DM** algorithms, and may be considered as a supporting service to assist in the process of algorithm development and validation (**AD&V**).

1.7 Virtual Products **[VP]**

This functional component consists of two major capabilities: (1) the ability to generate products "on demand;" and (2) the functionality required to implement a distributed archive and distributed data mining services.

1.7.1 Product Generation **[PRDG]**

This functionality provides standard and specialized products to subscribers and/or to other KBS algorithms and functions. If the product already exists (as the result, say, of the standard ingest process), then it is simply retrieved and shipped. More interesting, however, is when the requested product does not already exist, but must be created. The inputs to this process may be local or remote; and may be homogeneous or heterogeneous. For example, fusion algorithms may combine heterogeneous inputs into a single unified output. And the inputs to this fusion process may be held locally, or may reside at multiple distributed IA locations. In the latter case, they must be brought together, and the fusion algorithm applied. This example gives some indication of the type of processing this functional area provides. **PRDG** works closely with **DPRP** and with **FILT**, and may be called to supply inputs to other KBS algorithms and functions. It also relies on the distributed services provided by **PTP**, and the production it initiates will be under the supervision and optimization of **SC&A**.

1.7.2 Peer-to-Peer Services **[PTP]**

This is one of the most complex, and important, functional components of the system. It maintains all of the "distributed" aspects of the archiving function by co-operating with its peer IAs. This includes such features as: global search and retrieval (via globally maintained metadata and catalogs); distributed data mining; and distributed product generation. We have placed **PTP** within the **VP** functional component because "product" is a unifying theme to the distributed aspect of the IA. The idea is to shield users, and user processes, from details about (1) the location of input data sources and (2) the algorithms and models that are used to generate the desired products. But this abstraction requires exactly the type of global search capability and distributed processing provided by **PTP**. **PTP** also works closely with **OPT** to determine where, within the GRID, a distributed computation (be it data mining or product generation) should best take place. Current and predicted loading across the group of peer IAs is one of the key inputs to this optimization.

1.8 Data Access **[DA]**

1.8.1 Long Term Storage **[STOR]**

A key service provided by any archive, intelligent or otherwise.

1.8.2 Metadata Data Base System **[MDBS]**

Search and library services (see §1.9 below).

1.8.3 High Performance Disk Cache **[CACH]**

A key aspect of overall IA performance will be to optimize the use of the Disk Cache. This includes such strategies as pre-fetch of data sets based on predictive usage models (**OPT**), and co-operative participation in distributed production and data mining (**PTP**). The throughput of any KBS algorithm that reads or writes large volumes of data to **STOR** will be strongly affected by optimization of the **CACH** services.

1.9 Metadata and Libraries

In this section, we will briefly describe nine important data repositories and the role of each in overall system functionality.

1.9.1 Archive Metadata

This amounts to the searchable catalog of data sets and products currently held by the IA in STOR. This catalog is also part of a distributed globally searchable virtual catalog that includes the corresponding holdings of other peer IAs.

1.9.2 System Status

This data structure is updated by **STAT**, and used primarily by **OPT**. It contains both current and predicted status and loading of system hardware and software.

1.9.3 System Logs

This data structure is the history of system usage. It is maintained by **STAT**, and it can be "mined" by **STAT** to produce predictive models of system loading correlated to a variety of events.

1.9.4 User Profiles

This is where system security data structures reside: authentication, user access privileges, system time and storage thresholds, etc.

1.9.5 Algorithm and Application Libraries

A repository for the KBS and production software.

1.9.6 Algorithm Metadata

OPT will need to have performance models for the various **DM** and other algorithms it may be called upon to schedule. The performance models and associated parameters reside in this structure. When an algorithm is ready to be registered, as end product of the **AD&V** process, this data structure is updated to reflect the algorithm's performance characteristics and hardware requirements.

1.9.7 Product Metadata

As described in §1.6.1, the pedigree of the various products resident in the archive must be maintained and automatically verified to ensure comparability between and among data sources as well as compatibility with KBS algorithms and modeling software. This data structure houses that information, as prepared and written by **PMD**.

1.9.8 Science Models

A vetted library for use by DM and KBS algorithms and processes.

1.9.9 System Models

System performance models for use by system control and optimization algorithms and processes.

1.10. Internal Users

1.10.1 Administration

These users run daily operations. They monitor system status, control user profiles and privileges, respond to failures, install new hardware or software entities, and generally ensure smooth day to day system functioning.

1.10.2 Model Development and Validation

These users work collaboratively with application and science partners to develop and install new KBS models and algorithms. The **AD&V** functional component is the principal place within the system where this activity occurs.

1.10.3 Policy and Priority

These users set over-all system priority based on a balance of science goals and support for Application Partners. For example, of the wide variety of **DM** tasks that could be run in the background, which will be selected for execution? Or, of the wide variety of **DM** algorithms and/or research projects that might consume **AD&V** resources, which will be selected for active development?

1.11 External Users

1.11.1 Sensor Input

The high performance network interface to the real-time input data stream from the sensors and/or their Level 1 processes.

1.11.2 Sensor Web

The IA is one component of a network of sensors able to dynamically retask in response to events or collection opportunities detected by one of its participating nodes.

1.11.3 Interactive Users

Individuals or organizations that interface to the IA via the internet for tasking, search, algorithm development, and science.

1.11.4 Application Partner Decision Support Systems

ADP equipment operated by Application Partners capable of receiving reports from the IA and responding with requests for IA services (via **APIF**).

1.11.5 Peer Intelligent Archives

The IA is one of a group of peers that share data and services, and that co-operate in the optimized execution of distributed KBS algorithms and product generation (see **PTP** above).

1.11.6 Product Subscribers

Organizations and their systems capable of receiving IA products at high bandwidth.

1.12 IS/IDU Research Projects and Algorithms Applicability to the KBS/IA

	BROD	EMER	KARG_1	KARG_2	KUMAR	LEMGN	NEMANI	PAVEL	SMLYN	TENG	TILTON
1. SC&A											
1.1 STAT	X			X				X			
1.2 OPT	X						X	X			
1.3 AD&V	X		X	X		X		X		X	
2. IE											
2.1 IMPT							X				
2.2 EXPT							X				
2.3 WWW							X				
3. DM											
3.1 DPRP						X					
3.2 XFM		X									X
3.3 I&KB	X	X	X	X	X	X		X		X	X
4. ED											
4.1 IDS	X	X		X				X	X		X
4.2 CBR	X	X						X			X
5. FB											
5.1 SWIF						X	X				
5.2 APDS						X	X				
5.3 SP	X							X	X		
6. DQ											
6.1 PMD	X	X	X	X	X	X	X	X			X
6.2 FILT	X			X				X			
6.3 DMS										X	
7. VP											
7.1 PTP			X	X	X	X	X				
7.2 PG	X				X	X	X	X			
8. DA											
8.1 STOR		X			X						X
8.2 MDBS											
8.3 CACH					X		X		X		

IS/IDU Research Applicability

In this section, we will consider the role that advanced KBS algorithms and associated technology might play in a KBS/IA. Our starting point is the set of research projects studied at some length in previous work [9]. However, in addition to the nine research projects surveyed there, we will also include two addition projects: the work of Tilton, and of Emerson. As we will see, both of these fit well into the overall IA conceptual specimen architecture.

The table above summarizes the results of this section. The research projects label the columns, and the rows are labeled by the functional components and sub-components from the FDB. An entry in the matrix represents a potential for the associated functional components (row) to make effective use of the associated IS/IDU research (column). In the following subsections, we will provide additional discussion of the various contributions each project might play. That is, we have in effect sliced the matrix column-wise, and for each functional component, we will indicate how the research area might contribute to that functionality.

1.12.1 Brodley **[BROD]**

Brodley has investigated boot-strap approaches to classification and clustering. Thus, her algorithms do not depend on the availability of training sets, and derive prospective classes (and associated classification rules) without relying on domain models. The classification algorithms produced in this way can then be inspected by knowledge experts to see if the classes generated by the algorithm have an interesting scientific or economic underlying cause which would be of interest in its own right. If so, the classification scheme can then be refined, and used for a variety of possible applications.

Within **SC&A**, both **STAT** and **OPT** could benefit from classification schemes that identify trends in time series data. Both rely on the availability of good predictive models for system utilization (i.e., data mining of the archive usage logs) – something which **BROD** should be well-suited to provide. **AD&V** could benefit from the fact that **BROD** does not require training sets. This means that, in developing training data for other **DM** algorithms, a **BROD**-based algorithm could potentially kick-start the development effort by automatically extracting data sets of interest – a kind of automatically self-generated content-based retrieval.

I&KB is a generic place-holder for all sorts of KBS algorithms, of which **BROD** is certainly one. We have already touched on the fact that **BROD** can support a kind of boot-strapped **CBR**; and executing classification algorithms (like one produced by **BROD**) is at the heart of input data stream event detection (**IDS**).

BROD algorithms will certainly need to be modeled using **PMD** data structures. Using a **BROD**-based classifier for initial product error screening of input sensor data (**SP**) is a possible application. The strong dependency of **FILT** on **ED** means that **BROD** should be equally applicable to **FILT** functionality.

1.12.2 Emerson **[EMER]**

This research was not documented in the previous applicability study, but should play an important role in a KBS/IA. The Emerson technology computes (and stores both as data and metadata) the fractal dimension and associated geostatistical characterizations of underlying imagery. This provides an alternative representation – a transform – of the input data set that is independent of any particular science application that might choose to use it (e.g., for data mining purposes). This algorithm is well-suited for executing more-or-less unattended for long periods of time on very large portions of the archive. The results (and searchable metadata) are then stored with association back to the input sources, and can themselves now be inputs for a variety of other KBS techniques.

Along with **TILT**, then, **EMER** is a prime example of a **XFM**-style algorithm, and one of those accessible as a generic **DM** work-horse (**I&KB**). **EMER** could also be used as one step (e.g., a data conditioning algorithm) in an **ED**-style near-real-time data flow (supporting either **IDS** or **CBR**). The fact that **EMER** will often be used in back-ground mode over very long periods of time (months) on very large sets of data means that metadata describing the production pedigree of the outputs will be important (**PMD**) as well as optimized strategies for accessing long-term storage (**STOR**).

1.12.3 Kargupta – Distributed Data Mining **[KARG_1]**

This is the first of two research areas within the Kargupta team. This one references work performed on implementing distributed versions of standard large-scale data mining techniques. Since the IA resides in a distributed environment with multiple IA peers, this is an important and potentially very useful capability.

The development and validation effort for this type of processing is challenging, especially when more than one IA is to participate in a production run (**AD&V**). As a general purpose **DM** work-horse, **KARG_1** algorithms reside in and contribute to **I&KB** functionality. And the distributed nature of the processing means a strong connection to and role within **PTP** functionality. The production pedigree for products resulting from **KARG_1** algorithms can be challenging, because of the very large number of combinations of possible inputs and distributed processing contributions (**PRD**).

1.12.4 Kargupta – Distributed Mining on Real-time Streams **[KARG_2]**

This research direction has a number of operational characteristics (e.g., near-real-time time series analysis and signal processing) that suit it to collections of co-operating embedded sensors/actuators. Finding possible roles for this technology in an IA is challenging. However, we can propose the following possible uses.

Within the IA itself, monitoring and optimizing the utilization of the computing resources (processors, storage, networking, etc.) is a near-real-time function that might be able to utilize **KARG_2** functionality (**STAT** and **OPT**). However, it might be employed, it seems to fit best into a data flow mode of operations, characteristic of **IDS**; and this, in turn, could support **FILT**. As a distributed technique, **PTP** relationships will be critical, and generic relationships to **I&KB** and **PRDG** appear to be applicable.

1.12.5 Kumar **[KUMAR]**

As discussed at some length in the applicability study [9], of all the IS/IDU projects Kumar represents the best example of actually doing earth science using **DM** techniques (in this case, clustering using both spatial and temporal correlations over very large geographic regions). This means it has a role in **I&KB** that is different, and more important, than generic **DM** functionality. It becomes an exemplar for how science can be done using the KBS/IA capabilities. The extraction and conditioning of distributed data (**PRDG** and **PTP**) are key steps in the **KUMAR** approach; and because of the fact that **KUMAR** is mining time series data, there are embedded in it data management techniques that will be useful for both **STOR** and **CACH**. In Use Case 3 (§3.3 below), we examine these issues in greater detail for a **KUMAR**-like application.

1.12.6 LeMoigne **[LEMGN]**

This is the best example, among the IS/ISU researchers, of a fusion algorithm – one that accepts two (or more) compatible data sets as input, and produces a single composite output. A good example of its utility is in **DPRP** (preparing raw data sets, via value-added processing, into more usable, higher-level representations). The fact of multiple independent (but compatible) data sources means that **LEMGN** can both utilize and contribute to distributed **PTP** and **PRDG** techniques. This is also the first time we have seen a strong potential contribution to the sensor web and application DSS (**SWIF**, **APDS**) interfaces – by providing the ability to generate composite representation that enable detection and response in a feed-back loop scenario. And the need for strong pedigree tracking – both of the algorithm and the input sources – comes with this type of processing (**PMD**).

1.12.7 Nemani **[NEMANI]**

This research does not so much concern **DM** algorithms as processing architectures capable of automatically assembling lower-level products of algorithms, via work flow models, into finished user-oriented displays. As such, it focuses on architectural issues: optimizing the work; assembling distributed and or heterogeneous components; efficiently managing the computing resources, etc. This tends to place **NEMANI** in regions of the FDB that are mostly neglected by other researchers: **IMPT**, **EXPT**, **WWW**, **OPT**, **SWIF**, and **APDS**. The distributed aspects of the **NEMANI** approach could contribute to both **PRDG** and **PTP**; and performance optimization clearly will touch use of **CACH**.

1.12.8 Pavel **[PAVEL]**

While differing in technical details, at the application level both **PAVEL** and **BROD** share the same functional space, with many of the same objectives. The application domains for **PAVEL** include face and speech recognition; but the type of result returned by his unsupervised clustering approach is very similar, at an abstract level, to that of

BROD. Hence the strong overlap of their functional applicability, including: all functional sub-components in **SC&A** and **ED** as well as **PRDG**, **PMD**, **FILT**, **SP**, and (of course) **I&KB**.

1.12.9 Smelyanskiy **[SMLYN]**

Like **KARG_2**, the technology is best applied to near-real-time time series or signal processing data. As such, it does not fit well into the KBS/IA conceptual specimen architecture presented here. Its ability to handle high data rates means that there is an affinity and/or potential applicability to the **CACH** component, and (for similar reasons) to the **IDS** component (that is, it might be able to sit on the ingest stream and perform non-linear correlations or similar matched filter processing). We also speculate that it might be able to perform specialized analysis, in near real time, of sensor performance characteristics that might be useful as part of the **SP** feed-back process. The truth, however, is that **SMLYN** fits better at the part of the $O \rightarrow D \rightarrow I \rightarrow K$ food chain that is nearest to the sensor (e.g., in remote exploration scenarios and concepts of operation).

1.12.10 Teng **[TENG]**

This research, which has acquired the name of *data polishing*, is really a way of retraining classifiers by using modified training sets from which outliers have been "polished" away. Often the resulting version of the classifier is significantly more compact, faster, and more accurate than the unpolished version. Thus **TENG** is best suited to assisting in the development and optimization of other **DM** algorithms. The place reserved in the FBD for this functionality is data mining support (**DMS**) within the **DQ** component. Its major subscriber, **AD&V**, has also been called out, as well as **I&KB**, the generic locale for all **DM** algorithmic techniques.

1.12.11 Tilton **[TILTON]**

In terms of applicability, this algorithm is virtually indistinguishable from **EMER** – the other new research area included in this analysis. Whereas **EMER** transforms and extracts using fractal techniques, **TILTON** uses image segmentation based on geostatistical measures of high or low local variability. This representation does not require any science models, and (like **EMER**) can be thought of as a transform (**XFM**) of the data into a space where certain types of relationships, correlations, and features may be easier to detect. Thus, for example, long-term background processing of very large data sets is an attractive concept of operations, with the associated areas of applicability (**IDS**, **CBR**, and **STOR**).

2. Emerging Cyberinfrastructure

The purpose of this section is to review current or developing technologies that are likely to provide significant infrastructure support for any KBS/IA implementation.

2.1 Introduction

The cyberinfrastructure for a knowledge-based intelligent archive will enable users from a wide spectrum of research and application communities to access:

(1) high performance computing, networking, and communications;
(2) instruments that collecting data;
(3) data, information, and knowledge; and
(4) various services such as visualization, analysis and decision support.

Such a cyberinfrastructure will include organized and self-contained aggregates of resources, including instruments, technologies, information, services, and standard interfaces among different components. The components of such a cyberinfrastructure can be grouped into two categories: (1) a pure hardware-related category such as network bandwidth and computer hardware facilities; and (2) a middleware category which is primarily software technology built upon available hardware facilities. The focus of the following paragraphs will be on enabling architectures and technologies for the second of these, the middleware requirements.

2.2 Service Oriented Architecture and distributed services

The requests to an IA are diverse and dynamic, and may require dynamic chaining of multiple services available through different, and perhaps geographically distributed, service providers. The service-oriented architecture (SOA) is designed to address such requirements by constructing a distributed, dynamic, flexible, and re-configurable service system using internet protocols and technologies as the back-bone.

The key component in the service-oriented architecture is services. A service is a well-defined set of actions. It is self-contained, stateless, and does not depend on the state of other services. Stateless means that each time a requester interacts with a service, an action is performed. After the results of the service invocation have been returned, the action is finished. There is no assumption that subsequent invocations are associated with prior ones. In the service-oriented architecture, the description of a service is a specification of the messages that can be exchanged between the requester and the service. Standard-based individual services can then be chained together to solve complex tasks. The basic operations in SOA include publish, find, bind, and chain (Figure 2.1).

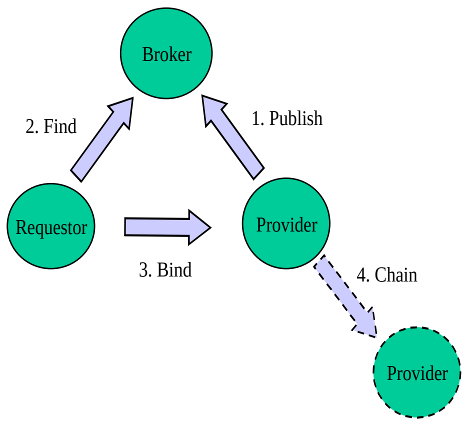


Figure 2.1 The basic service operations

There are three types of actors in SOA: (1) the providers who make specific services available over the Internet; (2) the requestors (users) who wish to access these services; and (3) the brokers who help requestors to find the right services. When a service provider sets up a service over the Internet for use by requestors, the service descriptions must be *published* to a broker (e.g., a registry, catalog or clearinghouse). When a requestor requests a service, the requestors and service brokers need to collaborate to *find* the right services. Service requestors describe the kinds of services they're looking for to the broker and the broker delivers the results that match the request back to the requestor. After the right service is found, a service requestor and a service provider negotiate as appropriate so the requestor can access and invoke services of the provider (*bind*). In many cases, a sequence of services must be bound together to produce the user-desired results (*chain*).

2.3 SOA and the Web

The SOA can be implemented using many different network environments. The two environments of interest here are the Web and the Grid. Web services mean services offered via the Web. The implementation of SOA in the web environment becomes Web services. A typical Web services scenario is that a client sends a request to a service provider at a given URL using certain protocol such as HTTP Get, POST, and SOAP. The service processes the request upon receipt and returns a response. A simple example of a Web service is a *data subset service* in which the client requests a data set within a user-specified spatial bounding box and temporal interval; the data service is then to return the data set that meets the client's specification. In this simple example, the request can be met almost immediately, with the request and response being parts of the same method call. A more complex scenario can involve chaining of different services provided by either the same or different providers. For example, a client wants to obtain a map at a given projected coordinate reference system. Services at the data provider site can provide data but not a map; and the data are at a different projected coordinate reference system from what was requested. In this case, the original request can only be met by chaining the data provider's service with a data coordinate re-projection service and a data-to-map conversion service. In this chain, responses (outputs) from one service may automatically be forwarded as operation requests (inputs) to the next service in the chain.

Web services are self-contained, self-describing, modular applications that can be published, located, and dynamically invoked across the Web. Web services perform functions, which can be anything from simple requests, as shown in examples in the previous paragraph, to complicated processes, such as extracting high level information or knowledge from a set of heterogeneous data sources. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service. Two characteristics provide the real power of web services:

- Everyone on the Internet can set up a web service to provide service to anyone who wants—many services will be available.

- The standard-based services can be chained together dynamically to solve complicated tasks – just in-time integration.

2.4 SOA and the Grid

2.4.1 Introduction

The Grid is a rapid developing technology originally motivated and supported from sciences and engineering requiring high-end computing, for sharing geographically distributed high-end computing resources ([10], [11], [12]). The vision of the Grid is to enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations ([11]). The key for the Grid success is the open source middleware called Globus Toolkit ([13], [14], [15]). It has become a de facto standard for all major Grid implementations. The implementation of services in Grid environment becomes the Grid services. Essentially, the Grid includes several types of middleware that can perform such tasks as computing and data system discovery and access, uniform security services, workflow management, fault management and recovery, large, complex, and distributed data archive management. With these middleware services, the Grid can provide on-demand, ubiquitous access to computing, data, and services and constructs new capabilities dynamically and transparently from distributed services.

The latest major version of Globus is Globus 3.0 [10], which implemented the Open Grid Service Architecture. The fundamental concepts of services in the Grid are the same as the Web services. However, they do have differences. A Web service can be invoked by any requestors over the Web while a Grid service can only be invoked by the requestors within the virtual organization. The web service practices also have been extended in Grid to accommodate the additional requirements of Grid services, including

- Statefull interactions between consumers and services
- Exposure of a web service's "publicly visible state"
- Access to (possibly large amounts of) identifiable data
- Service lifetime management

Currently the Grid and Web communities are converging through the Web Service Resource Framework (WSRF, [14]).

2.4.2 Lessons learned

Much research has been done on the development of the Grid and many lessons have been learned in different aspects of the Grid. These include:

- Initialization, installation, configuration, and operation are not easy tasks, including challenges in technical, organizational, and policy aspects.
- Software requirements are dependent on applications and on middleware used by applications.
- It is difficult to maintain consistency between different Grid sites, such as the stability and availability of resources, level of system dedication, level of expertise and area of interest of users/developers/administrators, version update of required environments and tools (e.g., Globus Toolkit).
- Close collaborations among discipline and computer scientists and among different communities are important but are sometime difficult.
- Standard infrastructure is very important, including interface, resources description and access.

2.5 SOA and interoperable standards

In order for SOA to work, interoperability standards related to all aspects of service operations are required. The major international bodies setting the web service standards are World-Wide Web Consortium ([22]) and Organization for the Advancement of Structured Information Standards ([18]). The body that sets the Grid service standards is the Global Grid Forum ([14]). The major standards related to Web services are shown in Figure 2.2. In addition to the general Web services, The Open Geospatial Consortium (OGC, formerly named OpenGIS Consortium) aims the development of interoperable standards for geospatial Web services implementation standards based on content and/or abstract standards set forth by various organizations including ISO, FGDC, and INCITS, as well as technology standards developed by W3C and OASIS.

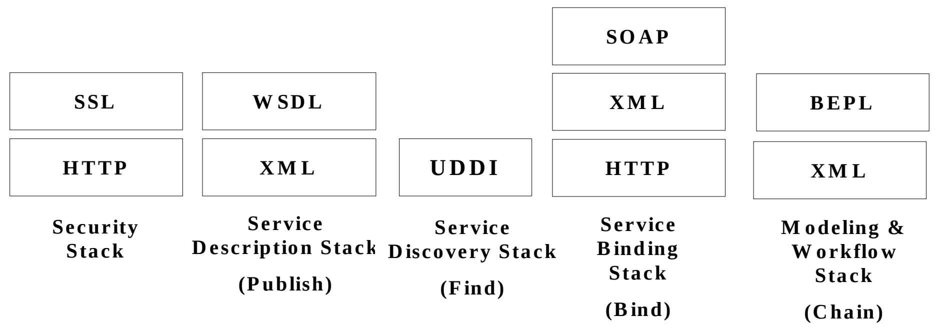


Figure 2.2 The major Web services standards

2.6 SOA and OGC technologies

2.6.1 Introduction

OGC has successfully executed several phases of the OGC Web Services (OWS) initiative, including Web Mapping Testbed I (WMT) phase I and phase II, and OWS 1.1, 1.2, and 2.0. These OGC activities were mainly intended to advance interoperable geospatial Web services technology, support development of multi-vendor portable demonstration, and feed requirements and recommendations into OGC's standard specification process. The activities helped to identify issues concerning the lack of interoperability among systems that process geospatial data, and helped to develop, test, and demonstrate consensus-based implementation specifications. Current status is that a set of Web-based data interoperability specifications has been established. These include:

- Web Mapping Specification (WMS), which allows interactively assembling maps from multiple servers;
- Web Coverage Service (WCS) Specification, which defines interface for accessing rater (grid, image) type geospatial data, especially those of remote sensing data, from multiple coverage servers;
- Web Feature Service (WFS), which defines interface for accessing feature (vector) type geospatial data;
- Web Registry Service (WRS), which specifies data and service publishing and discovery interface;
- Web Coordinate Transfer Service (WCTS), which defines interface for performing coordinate transformation of geospatial data; and
- Web Image Classification Service (WICS), which specifies interface for classifying image type data.

These specifications allow publishing, searching, and access to geospatial data and services in a distributed environment, regardless of data format, coordinate reference system, spatial/temporal resolution, and geographic locations of data archives and service providers.

2.6.2 Lessons learned

A number of OGC testbeds and initiatives had been successfully conducted in developing interoperable standard interfaces. Lessons learned from these activities include:

- Interoperable, standard compliant interface significantly improve the publishing, discovery, and access to geospatial data and services.

- Development of interoperable standard interface is required for Web services but formal adoption of such standard interfaces may take some time. Most OGC testbeds and initiatives were run at very fast tracks (e.g., six months). Several rounds of revisions and implementation tests/demonstration were usually needed to reach a stable specification. For example, the first version of the WCS specification, version 0.5, was completely tested in November, 2001. It was updated to versions 0.7, 0.8, 0.85, 0.86, 1.0.0, and now 1.0.20 as of September, 2004. Revisions will still be needed before it can be finally considered stable specification.

- Collaboration and consensus among different users, vendors, organizations, and communities are important to the successful development of standard interface. There are always ambiguities for clients and service providers to adopt OGC specifications, especially concerning terminologies, parameters, keywords used differently among different people. The process of reaching consensus can be lengthy and onerous.

3. Use Cases

In this section, we will present three use cases to illustrate the intended concept of operations for the Conceptual Specimen Architecture described in Section 1. Three use cases will be presented: (1) a use case based on near-real-time feed-back to an application partner, relying on event detection functionality (§3.1); (2) a data mining use case relying on on-going back ground processing using a data transformation algorithm (§3.2); and (3) a data mining use case focused on a specific research question in Earth Science (§3.3). A final section (§3.4) then discusses how the knowledge building technologies described for an Intelligent Archive are also applicable to remote exploration scenarios, such as manned or unmanned missions to the Moon or to Mars.

3.1 Use Case 1: Feed-back

At a top level, the functionality provided by the KBS/IA will perform automated event detection in near-real-time on an incoming data stream from an earth observing sensor (such as MODIS or AMSU). The event detection processing is incorporated into the standard ingest processing flow for geographical areas of interest; and the detection of an "event" (forest fire, infestation, algae plume, volcanic activity, etc.) is relayed to Application Partner automated decision support software for analysis and response. In the scenario we will present, the IA also provides the Application Partner with access to other data mining capabilities (if required), and to the Sensor Web to obtain additional observations in other spectra or at other resolutions.

We have broken the discussion into four areas: development, validation, and registration of the event detection algorithm (§3.1.1); optimization (§3.1.2); production (§3.1.3); and other capabilities and considerations (§3.1.4).

3.1.1 Development, validation, and registration of event detection algorithm

Before a KBS algorithm can be put into production, it must be validated, and parameters that characterize its performance characteristics must be determined. The existing data mining capabilities already in place can assist in this process. For example, suppose that a "high fidelity" science model for the phenomena already exists, but that its performance characteristics (in terms, say, of latency) do not meet the application requirements. In that case, it may be useful to construct a tuned pattern matching algorithmic implementation (say, using a neural net, or a support vector machine) that meets classification accuracy requirements but that executes with acceptable latency. The data mining facilities of the KBS/IA can extract training and test sets using the high fidelity model, which are then used as input to the Artificial Intelligence algorithms. This is an important aspect of the KBS/IA – one generation of technology leveraging the next.

An administrative process oversees the selection of candidate development efforts based on a prioritization of science needs, national policy, available resources, and risk. Staff within the KBS/IA assists scientists and Application Partner representatives to specify requirements, and the algorithm development process is then initiated. Functional components that might come into play, at this stage, include (see the FBD from Section 1):

> **AD&V**, to oversee the development and validation of the algorithm;
>
> **I&KB**, for inferring a model of the event from available data;
>
> **CBR**, for retrieving training data;
>
> **DMS**, to assist in algorithm refinement and performance enhancement; and
>
> **PTP/PRDG/DPRP**.

This last trio calls for some explanation. In order to automatically detect the event of interest, it may be necessary to perform some preliminary data transformation (**XFM**) or other processing, including as input data sources derived from peers in the distributed network of KBS/IAs. Such preprocessing might include, for example, fusion of multiple datasets into a single co-registered dataset. The working out of the specific form of the required inputs, and arranging for them to be available from distributed peers, is to be carried out by sub-components within **VP** with the participation of **DPRP**.

Models for the computational performance of the **ED** algorithm under development – throughput, access to secondary storage or disk cache, memory, etc. – must also be developed for use by the system optimization algorithms (in **OPT**) that oversee actual production (for entry into the searchable *Algorithm Metadata* structures, §1.9.6).

3.1.2 Optimization

This portion of the use case, while conceptually distinct, has significant overlap with both the development (§3.1.1) and production (§3.1.3) stages. The task, here, is to ensure that the processing flow implementing the algorithm is optimized with respect to a number of criteria: production priorities; physical data access (i.e., use of the high performance data cache, pre-fetch, etc.); and production flow dependencies (so that work required by multiple ingest stages need only be performed once). In the use case under consideration, the **ED** algorithm is to be incorporated into the ingest production flow, via **IDS**. Where, within that flow, to place the detector, and how to route the output (including any metadata and logging data the **ED** algorithm may produce) is also part of the optimization process.

There may also need to be negotiations with KBS/IA peers, if inputs from them are required. Or, *mutatis mutandis*, it may be that a feed-back process from a distributed peer requires input, and the production resources to provide it would then need to be made available (subject to global prioritization policy). It may also be appropriate to develop stochastic models for uncertain events: the likelihood of event occurrence, the false alarm rate, the expected level of processing required should an event be detected and verified (including increased user access to KBS/IA stores and products resulting from the scientific interest in the event), etc.

These decisions, or production models, are the responsibility of **OPT**, using especially the performance models held in Algorithm Metadata, and the services of **VP/PTP**.

3.1.3 Production

At this point, the **ED** algorithm is ready to be placed in the ingest production stream, via **IDS**. If we assume (as seems reasonable) that interest will be focused on relatively small geographical regions (e.g., north western Wyoming), then sensor access patterns to that region can be used as triggers for implementing this special functionality, which need not be continuously applied to all the incoming data. It is also likely, in this type of scenario, that the algorithm will make use of change detection. This means that some amount of history (short, if change detection is highly dynamic; or more lengthy, if trends over time are to be used) must be available – presumably, via pre-fetch into the high performance data cache. And as noted above, it may also be required to access and fuse remotely held related data, depending on algorithmic requirements.

When a possible event occurs (with some known false alarm rate), a sequence of resulting actions is then set in motion. The **AP/DSS** is notified, together with whatever associated data products may be required. This DSS then has available a number of options. It may decide, using its own algorithms and/or expert review, that the event was a false alarm. Or, it may request additional **DM** services from the KBS/IA (e.g., enhanced products or virtual products). Or, it may request dynamic retasking, via **FB/SWIF**, of resources in the Sensor Web (assuming it does not have direct, independent access of its own). Or, finally, it may choose to change the operational characterics of the **ED** algorithm: changes to thresholds, or to production priority, etc. This input from the AP/DSS to the KBS/IA is critical to providing optimized dynamic response based on the actual properties of the event under consideration.

3.1.4 Other capabilities and considerations

During production, **STAT** is keeping track of the resources used, and access patterns resulting from the detection of an event. These are entered into the *System Logs*, and then become available for use by **DM** algorithms looking for usage and loading patterns. The idea is that **OPT** can then use this type of data to improve its own stochastic models of system utilization – e.g., by pre-fetching historical data in the area of interest into **CACH** to be ready to respond to anticipated user interest.

Of course, in the background are all the other on-going functions of the KBS/IA: the normal ingest procedures, for example, including updating of metadata, execution of required **XFM** algorithms on the input stream, transferal of data to **STOR**, etc. Part of the job of **OPT** is to ensure that the **ED** functionality does not interfere with these ongoing production requirements, and to optimize the flow in response to dynamic occurrences consistent with administrative policy and priorities. As we mentioned briefly in passing, it may also be necessary to support the **ED/FB** requirements of a peer, and the associated loading on the local system (again, partly deterministic based on known sensor access patterns, and partly stochastic based on uncertain event occurrence) should be modeled and enter into optimization calculations.

3.2 Use Case 2: Background data mining using **XFM**

At a top level, a data transform algorithm is applied to large selections of the underlying data archive. Typically (despite the notional flow in the FBD), such algorithms accept as input *data* from the $O \rightarrow D \rightarrow I \rightarrow K$ KBS hierarchy, and produce either *data* (of a transformed kind) or *information* as output, along with searchable descriptive metadata. As will become clear during our discussion, such a transform is typically self-contained (that is, it relies on nothing other than an input data entity) and is not focused on any particular science problem (although the solution of science problems, or the building of science models, may choose, if they wish, to utilize the transformed products). As such, a transform is well-suited to implementation as a background process operating over an extended period of time – up to many weeks, or months – in a prioritized order over very large subsets of the archive.

As above, we have broken the discussion into four areas: registration of the transform algorithm (§3.2.1); optimization (§3.2.2); production (§3.2.3); and other capabilities and considerations (§3.2.4).

3.2.1 Registration of the transform algorithm

Unlike data mining algorithms (discussed in §3.1 above, and §3.3 below), transform algorithms are typically developed independently of the archive, and cannot make use of its data mining services (other than to provide examples of input data sets on which to test the code). As a result, it is likely that a transform comes to the KBS/IA in close to finished form, and need only be incorporated into the production libraries. This means building a performance model (for use by the optimizer) as well as review by the administrative oversight function that establishes production priorities. This review must decide such matters as: which transform algorithms to execute, with what relative priority, on which subsets of the archive, and in what order. Given the priority scheme (in our envisioned use case, the overall priority is low, so that **XFM** operates as an ongoing background process, consuming otherwise unused cycles) and the order in which to process the target data sets, **OPT** can now schedule the **XFM** functionality.

The output of such a transform can, itself, be considered as a kind of product (typically at the *data* or *information* KBS levels). In that case, it can be: (1) pre-computed (the case of interest here, by **XFM**); (2) a virtual product (that, computed on-demand by **VP**); or (3) computed as part of the ongoing **ED** or **IMPT** functional components. In practice, the metadata that is produced as an auxillary to the transform is critical to utility, since the transformed product itself may be nearly as large as the underlying data set from which it was produced. From a processing perspective, these algorithms are typically highly parallelizable, both internal to the algorithm and externally, across the independent input data entities. As such, they may be able to make very effective use of special purpose high performance parallel hardware (MIMD, SIMD).

3.2.2 Optimization

Given a performance model, priority, and order for processing, **OPT** is well-placed to make efficient processing scheduling decisions. In addition, we note two other factors **OPT** may wish to take into account. First is the *physical data model* for the output data sets. This is a consideration for any product to be permanently placed in the archive, and it depends on having some notion of how the data is intended to be used. The point is that to access data by (say) time series at a single location is a very different operation from access to data covering a large area but that was collected at a single time. How the data is stored will determine, in large measure, how long a seek-and-retrieve operation will take; and it may be possible for **OPT** to arrange these matters in advance (or to provide multiple versions) to as to speed the **DM** application that will eventually access the data.

This, in turn, bears on another performance enhancement – namely, the staging of the data to (on input) and from (on output) **CACH**. Again, **OPT** may be able to make decisions that improve throughput, and that reduce the likelihood of conflict with other ongoing processes (which may be subject to uncertain stochastic events requiring dynamic reoptimization).

3.2.3 Production

The production strategy considered in this use case is simple: consume available unused cycles to complete, over time, and as a background process, the transformation of specified large subsets of the archive.

As mentioned briefly above, there are other production modes in which **XFM**-style algorithms could be used, and we pause here to mention three of the most important. First, it is possible to impose **XFM** algorithms as part of **IMPT** and **IDS** functionality. In the case of **IMPT**, not only the data received from the sensor would be stored (along with its metadata); but also the transformed version of the data (along with its separate metadata). Thus, from a single input data stream there may arise two or more output data streams for entry into **STOR**. Second, it is possible to utilize an **XFM** algorithm as part of **ED** – for example, as discussed in §3.1, in support of **FB** functionality. Here, the product of **XFM** is immediately consumed, and is not stored for later retrieval. And third, it is possible that **XFM** might be invoked as part of **VP** functionality, either returning an on-demand product to a local user, or in support of a user process located at a distributed peer in the KBS/IA network.

Another important aspect of production functionality is support for **PMD**. Recall that this is the function that keeps track of the production pedigree of the various products stored in the archive, and that ensures that using processes are compatible (in terms of their input assumptions) with the version and formats of the algorithms used to create the data sets. Since **XFM** used in this mode creates a “product” in this sense, it must furnish the corresponding descriptive metadata to **PMD** for entry into *Product Metadata*.

3.2.4 Other capabilities and considerations

As mentioned briefly above, the algorithmic characteristics of **XFM**-style algorithms make them ideally suited to special purpose massively parallel hardware (MIMD or SIMD). Ideally, the KBS/IA performance models (in *Algorithm Metadata*) will be able to capture this aspect of these algorithms and, if such hardware is available, make suitable use of it.

3.3 Use Case 3: Large scale data mining for science research

We begin by observing that the discussion of this use case is made difficult due to the very large number of different approaches a research scientist may take in the investigation of a problem or phenomenon of interest. Of these, we mention two in particular with significant impact on the data mining technologies that might be employed. First, there are the so-called “boot-strap” methods which attempt, via clustering or similar aggregation approaches, to identify statistically significant patterns or relationships in large data sources which had not been previously noticed. As an alternative, we may cite the work of Kumar [16], in which a type of relationship which was already known (an “index”) was used to guide development of a sophisticated temperal-geographic classifier. This classifier was then able to identify other previously undetected instances for further study. In the discussion which follows, we will have Kumar’s work in mind as a general model; but we will also attempt to indicate important variations or excursions which might come into play if the research goals or application domain were somewhat different.

As with the first two use cases, we separate the discussion into four partially over-lapping general areas: development, validation, and registration of the data mining algorithm (§3.3.1); optimization (§3.3.2); production (§3.3.3); and other capabilities and considerations (§3.4.4).

3.3.1 Development, validation, and registration of the data mining algorithm

As indicated in the discussion above, an important initial question concerns whether there already exist examples of the phenomena of interest, or high-fidelity models which could be used to search for such examples. If so, the general techniques already described in §3.3.1 above would be appropriate here, as well. If not, then boot-strapping methods must be used. These can be very human expert intensive, since a clustering algorithm (to use the most common approach as an example) will often extract “relationships” that are not causal but simply reflect the statistical properties of the particular example that is under examination. Thus, a review of the product by an expert is required; and this review is typically the gating factor controlling overall throughput for this stage of the end-to-end process.

In Kumar's work, this stage was able to rely on existing examples of "indices" to use guide the development of a clustering algorithm so as to be able to extract and hence reproduce the known phenomenon. The resulting classifier could then be applied to new data to see if similar previously unknown instances might emerge.

Once a relation has been identified as having scientific merit (and here, the administrative review and prioritization process comes into play), the algorithm to detect its presence can be honed to achieve acceptable throughput as well as Type I and Type II error rates. The KBS/IA can be used to generate test sets for validating these requirements (**CBR**), and for generating the algorithm performance models for use by **OPT**.

3.3.2 Optimization

Because of the greater complexity of this problem when compared to our two previous use cases, the optimizer may need to take a number of additional factors into account. First, there is the issue of the physical architecture of the archived data, and the match (or lack of it) between this physical architecture and the access patterns the algorithm will require. One important example of this (which we mentioned in passing above) is the difference between accessing wide geographic areas for which the sensor data was collected fairly close together in time, on the one hand; versus a number of time collections, spread over a considerable length of time, but for a narrowly circumscribed area. The physical data architectures (that is, the position of the data sets on the tape or spindle) to optimize these two types of retrieval are very different, and it will fall to the optimizer attempt to understand these performance constraints and impacts and, where possible, perform physical rearrangement (or pre-fetch) of the data to keep overall throughput at acceptable levels. Optimizing the use of **CACH** will therefore form a significant part of structuring the process flow.

It is also much more likely, when compared to the previously discussed use cases, that the distributed capabilities of the network of KBS/IA will be required – either through the assembly and preparation of the data sets (**VP/PTP/PRDG/DPRP**), or through the direct implementation of distributed data mining algorithms (e.g., as per Kargupta [9, §3.5]). Data quality issues are also likely to arise here, particularly with regard to the processing pedigree of the input data sets (see **PMD** and *Product Metadata*). The form of the output, and metadata descriptions of it, also affect these two entities.

The full end-to-end process will probably also require feed-back and iteration, as additional runs are required based on new discoveries and/or questions arising from earlier runs. Hence, ongoing administrative review of the overall progress of the research, with possible adjustments to priority, must be provided.

3.3.3 Production

The computational demands generated by this type of research can vary enormously, depending on the complexity of the correlations being investigated, the size and resolution of the data, the fidelity of models embedded in the algorithms, etc. Runs can vary from near-interactive rates, on the one hand, to long term batch or back ground processing, on the other. Depending on the amount and characteristics of the output products, there may be a need for specialized analysis and display software (e.g., 3-D immersive virtual reality displays). And as touched upon briefly above, we envision an ongoing assessment of the scientific merit of the results that are being produced, and the ability to adjust production and resource priority dynamically in response to stochastic shifts in system loading as well as scientific merit and risk of the research.

Since the KBS/IA is one peer in a network, it will also be called upon to support the **DM** activities initiated at remote sites. This, in turn, may result in additional loading on the local resources (CPU, memory, disc, network bandwidth, etc.), and this loading must be anticipated (when possible, using predictive models) and factored into optimized dynamic adjustments to processing flows and relative priorities (**OPT/PTP/I&KB**)

3.3.4 Other capabilities and considerations

The use of the type of data mining capabilities such a KBS/IA could provide is still far from routine in the Earth Science community, and it is still unclear, of the many capabilities that could be provided, which will prove to be most useful in the long run. Faced with this uncertainty, it seems best to focus first on a few areas which appear to have the greatest possibility of success. By showing value on well-chosen demonstration examples, the utility of the KBS/IA concept can be shown; and this can then serve as a basis for incremental expansion to the inclusion of additional capabilities.

It should also be observed that the **DM** activities described here will often occur during the first stage of other functional components. For example, an **ED** algorithm may be required; but the *development* of the algorithm may entail the use of sophisticated **DM** and **I&KB** activities. As noted at the outset, there are technology dependencies, and capabilities refined in one area can then be enablers for advances or improvements in a different but related area.

3.4 Use cases involving remote exploration

In this section we will consider another application domain for KBS systems and technology: remote exploration. In previous work [9], we indicated how the KBS algorithms developed under the IS/IDU program were applicable beyond Intelligent Archives. The brief discussion here will extend those ideas, to show that not only the algorithms, but the architectural principles also extend to this other application domain.

A key feature of the top level FBD is that the archive accepts from the sensor an input data stream which is processed, including perhaps event detection or data transformation, prior to storage. If we think of the variety of both embedded and mobile sensors required to support remote exploration, the architectural analogy comes into focus. The biggest difference between sensor input streams for the RE application, on the one hand, and for the IA, on the other, is that the sampling rate of the supported sensors is very different:

on the order of hours between co-located samples for an Earth Observing sensor supported by an IA, on the one hand; and perhaps 10's to 100's of Hz for embedded sensors in an RE scenario, on the other. Hence, the focus of the intelligent algorithms will shift from change detection and pattern recognition within imagery to something much more like signal processing. But the idea of a filter sitting on and examining an input stream, and extracting from it information concerning data quality, event detection, and higher-level summaries (i.e., metadata) is the same in both cases. Indeed, some algorithms (**SMOL**, **KARG_2**) which have a difficult time fitting into KBS/IA use cases have a natural fit in RE scenarios.

A second analogy derives from the distributed nature of the required processing. In the KBS/IA use cases, we saw how peers in a network of co-operating IAs could share not only data but processing resources in an optimal way to satisfy user requests for products (**VP**) and **DM** services. The analogy in a RE scenario is co-operating teams of semi-autonomous sensors or other agents. If we think, for example, of a team of a few (7 to 20) mobile sensors communicating as they jointly perform a surveillance task, the need for distributed algorithms, optimized role assignments, and dynamic adjustment in response to uncertain events is clear.

As just suggested, the control algorithms to optimize resource utilization governed by policy and priority also carries over into the RE domain. Model Predictive Control and its associated modeling and state estimation requirements is equally applicable to both types of use case. In the case of co-operating teams introduced above, important objective functions requiring near-real-time dynamic reoptimization include: maximizing the value of collected information; maximizing use of available bandwidth back to a centralized controller; minimizing risk; assignment and reassignment of roles in response to agent failures or degradation.

The monitoring functions in an IA – data quality assessment, event detection on the input data stream, state estimation and system status – also have a strong analogy in RE use cases. Examples include: health and status (e.g., habitat monitoring); detection and avoidance of incipient failure modes; predictive models for space and solar weather; and optimized route planning based on learned (i.e., modeled) terrain information.

The conclusion from this brief discussion is that not only the IS/IDU research and resulting algorithms and technologies, but also the underlying architecture which unifies these components into a useful whole, extends in a natural way from the KBS/IA case studied in this paper to the KBS/RE case that is a high NASA priority.

4. Top-Level Software Architecture and Specimen Design

The purpose of this section is to briefly describe a candidat software architecture for a KBS/IA capable of efficiently implementing the functional requirements and use cases described in Sections 1 and 3 above. Together with Section 2, then, this section forms a bridge to the next stage of research: development of software prototypes and demonstrations.

The material is organized as follows. §4.1 introduces and motivates the selected approach by showing how it represents a natural evolution from, and enhancement of, current stand-alone archival technologies. In §4.2, the key components are described. In §4.3, the Service Oriented Architecture ideas described in Section 2 above are illustrated by means of a worked example based on **KUMAR** (see Use Case 3, §3.3 above). This is further elaborated in §4.4 by showing how this design can be described using the Web Ontology Language, OWL.

4.1 Top-Level Software Architecture and Specimen Design

4.1.1 Block Diagram of the Top-Level Software Architecture

This section discusses the top level software architecture of an intelligent archive (IA), which follows the concept of service oriented architecture as described in section 2.2. The discussion describes the evolution from a non-intelligent, traditional, and standalone archive to an intelligent archive which can either be centralized or distributed geographically.

4.1.1.1 The architecture of traditional archive with no intelligent data understanding

A traditional, standalone data archive architecture is shown in Figure 4.1. In this architecture, the archive provides data it archived to the user through the interface provided by the archive. The user usually first searches the archive catalog through the user interface and, upon finding desired data, requests and obtains data from archive. The three components, user interface, archived data, and the catalog, belong to the archive. From the user point of view, interactions with the archive for obtaining data are made available by the archive itself. When new data arrive, they are directly ingested and cataloged in the archive catalog. In such a traditional archive, the content and format of data available to users are fixed at the point in time when the data are ingested into the archive.

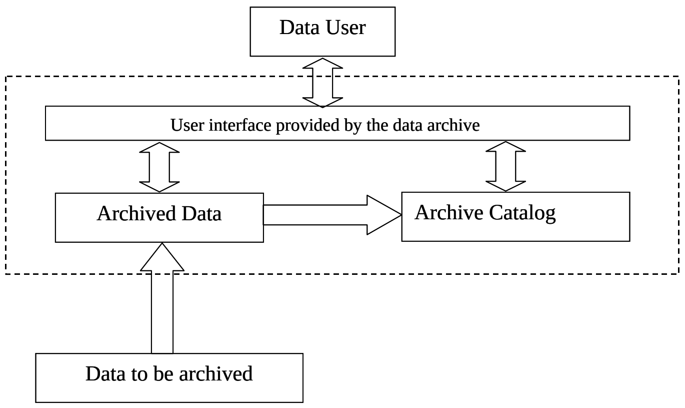


Figure 4.1 - Traditional archive architecture

4.1.1.2 The architecture of standalone archives with limited intelligent data understanding

Intelligent data understanding (IDU) can be built in to a traditional standalone data archives by imbedding IDU algorithms into the archive, as shown in Figure 4.2. In this architecture, algorithms can be run on the archived data and the outputs of the algorithms are archived as derived data, usually high level data products, information or knowledge, and cataloged in the archive's catalog. The algorithms built into the archive usually function independent of each other and work with only one or a few specific types of archived data. The interfaces between individual algorithms and archived data are algorithm-specific. There is no direct linkage between the algorithms and the catalog in the archive. In other words, users can access the IDU results archived in the archive storage but the algorithms themselves are not accessible to users. Such archives can well serve communities of similar disciplines where user requirements can be largely met with simple IDU algorithms.

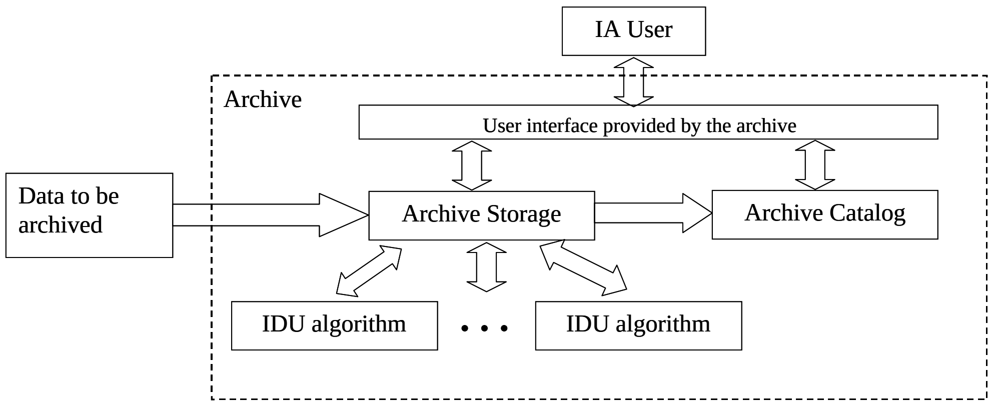


Figure 4-2 – Limited additional intelligence

4.1.1.3 The architecture multiple connected archives

Standalone data archives, such as has been shown in figure 4.2, either geographically distributed or centralized, can be linked together to construct more useful archive with more IDU algorithms and more content availability. Each connected archives are the members of this linked archive. In order to connect multiple member archives, improvements in the user interface are necessary so that users can access multiple member archives by means of a single common user interface. For example, if two such archives are connected, a common user interface that allows users to access storage and catalogs in both archives is desirable. To further improve the functionality of the archives, it will be desirable to build a common interface between the archive storage and the catalogs so that only one common catalog is necessary in the linked archive (Figure 4.3). The common catalog, of course, can be populated by member archives, but the content and the interface to the catalogs must shield the user from unwanted detail, so that there is a single common logial catalog for the entire ensemble.

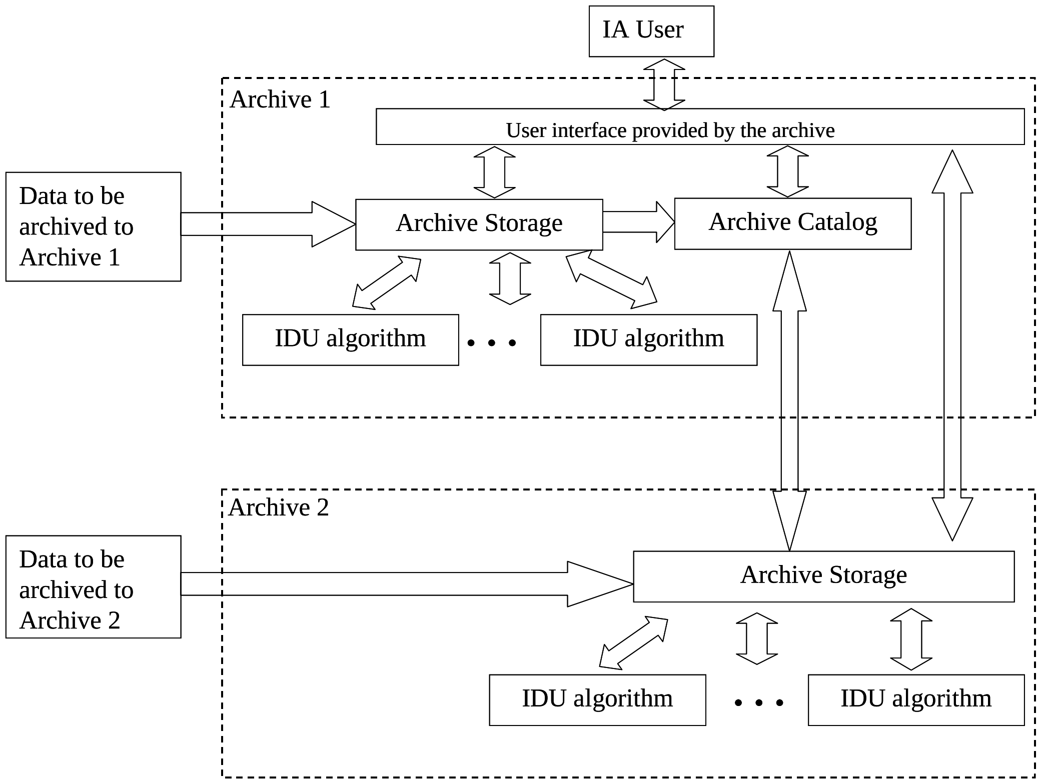


Figure 4.3 – Distributed Archives

4.1.1.4 The service oriented software architecture of knowledge building system and intelligent archive

Further improvements to multiple connected IAs can be achieved by building common interfaces not just between user and the archives but also between IDU algorithms and the archived data and between IDU algorithms and users. Based on Service Oriented Architecture (SOA), a knowledge building system/intelligent archive (KBS/IA) can be built with fundamental changes to architectures shown in Figures 4.1, 4.2, and 4.3. In such a new architecture individual archived data and individual algorithms are considered as completely independent components. Individual algorithms, as well as the catalog, are considered as services (Figure 4.4). In Figure 4.4, the IA is also modeled as a component in a complete data/information/knowledge system that includes sensor system and different levels of data product generation systems. Different interfaces in such a system are identified:

(1) a common service interface through which user interacts with the IA to request different services, including catalog service, through which users interact with the KBS/IA;

(2) a common storage interface through which services interact with archived storage;

(3) a user storage request interface through which users obtain stored contents (this interface, however, can be integrated into the common service interface, as will be discussed later);

(4) the feedback loop interface through which IA interacts with sensor system to perform things like sensor tasking and dynamic data collections; and

(5) the sensor system including sensor and low level data production providing data to the archive.

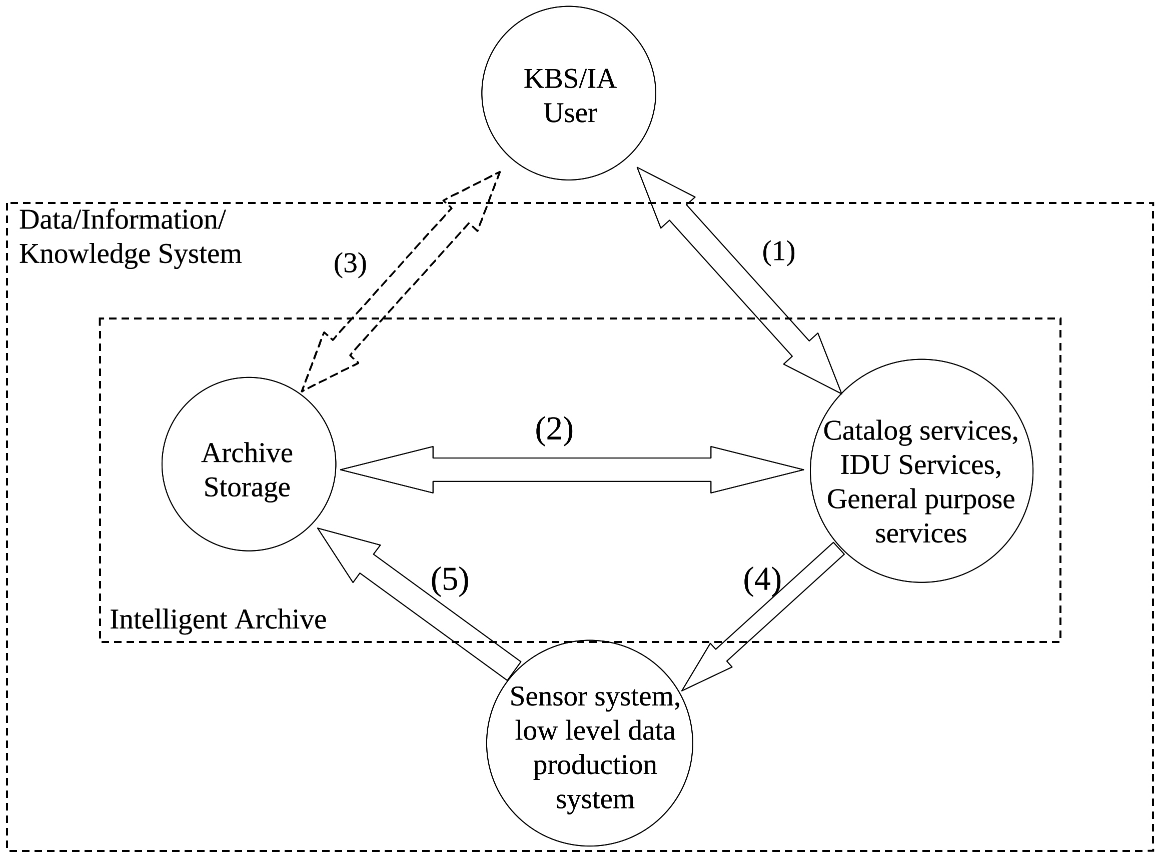

KBS/IA User
Data/Information/ Knowledge System
(3)
(1)
Archive Storage
(2)
Catalog services, IDU Services, General purpose services
(5)
(4)
Intelligent Archive
Sensor system, low level data production system

Figure 4.4 – A Service Oriented Architecture for the KBS/IA

The direct user storage request interface, (3) in Figure 4.4, is useful only when content (data, information, knowledge) requested by users is directly available in the archived storage without any need of services, such as a complete granule of data. For all other requests, going through this direct interface indicates that results of all services, including IA services and general purpose services, must be pushed back to the archive storage before users can obtain them. This, however, is not necessary because in many cases the output of services may not need to be archived, especially for general data processing service such as spatial subset and data format conversion. Further more, even in case of requesting a complete data granule, the user usually still needs to request the catalog service to search and locate the data granule first before he/she can actually request the data. Therefore, this interface should be included in the more general common service interface, i.e., interface (1).

4.1.1.5 Top-Level Software Architecture of KBS/IA and Specimen Design

The service oriented architecture shown in Figure 4.4 is at abstract level. A detailed top level architecture of a KBS/IA is given in Figure 4.5, where different components including the IA algorithms/service, archive storage, and catalog are shown. Figure 4.6 is yet a more detailed diagram of a KBS/IA in the context of specific IDU algorithms, i.e., Vipin Kumar's time series, clustering, and association algorithms ([16]; [19]; [23]; [17]). It also serves as a specimen design of the software architecture. Service/data flow sequences and input/output contents are identified in this diagram with directional links between different boxes. The components, interfaces, and flows are discussed in sections 4.2 and 4.3.

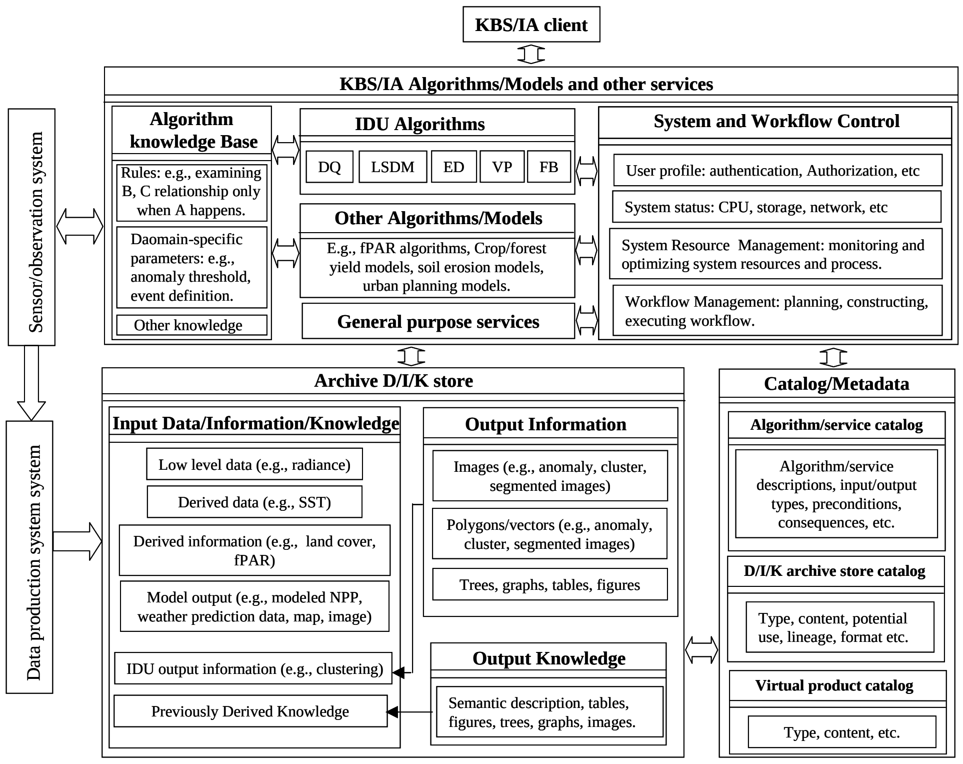

KBS/IA client
KBS/IA Algorithms/Models and other services
Algorithm knowledge Base
Rules: e.g., examining B, C relationship only when A happens.
Daomain-specific parameters: e.g., anomaly threshold, event definition.
Other knowledge
IDU Algorithms
DQ
LSDM
ED
VP
FB
Other Algorithms/Models
E.g., fPAR algorithms, Crop/forest yield models, soil erosion models, urban planning models.
General purpose services
System and Workflow Control
User profile: authentication, Authorization, etc
System status: CPU, storage, network, etc
System Resource Management: monitoring and optimizing system resources and process.
Workflow Management: planning, constructing, executing workflow.
Sensor/observation system
Data production system system
Archive D/I/K store
Input Data/Information/Knowledge
Low level data (e.g., radiance)
Derived data (e.g., SST)
Derived information (e.g., land cover, fPAR)
Model output (e.g., modeled NPP, weather prediction data, map, image)
IDU output information (e.g., clustering)
Previously Derived Knowledge
Output Information
Images (e.g., anomaly, cluster, segmented images)
Polygons/vectors (e.g., anomaly, cluster, segmented images)
Trees, graphs, tables, figures
Output Knowledge
Semantic description, tables, figures, trees, graphs, images.
Catalog/Metadata
Algorithm/service catalog
Algorithm/service descriptions, input/output types, preconditions, consequences, etc.
D/I/K archive store catalog
Type, content, potential use, lineage, format etc.
Virtual product catalog
Type, content, etc.

Figure 4.5 – Detailed Top Level Software Architecture for KBS/IA

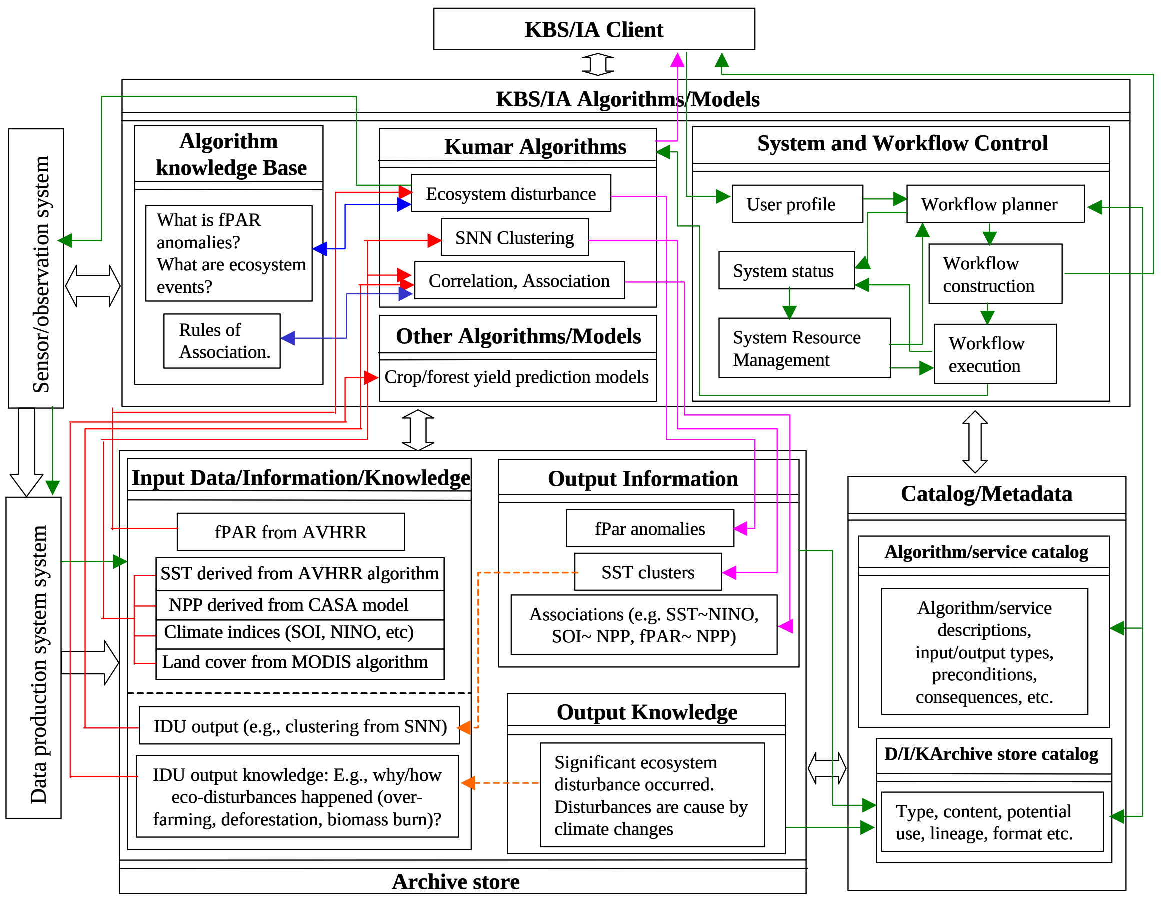

KBS/IA Client
KBS/IA Algorithms/Models
Algorithm knowledge Base
What is fPAR anomalies? What are ecosystem events?
Rules of Association.
Kumar Algorithms
Ecosystem disturbance
SNN Clustering
Correlation, Association
Other Algorithms/Models
Crop/forest yield prediction models
System and Workflow Control
User profile
Workflow planner
System status
Workflow construction
System Resource Management
Workflow execution
Sensor/observation system
Data production system system
Input Data/Information/Knowledge
fPAR from AVHRR
SST derived from AVHRR algorithm
NPP derived from CASA model
Climate indices (SOI, NINO, etc)
Land cover from MODIS algorithm
IDU output (e.g., clustering from SNN)
IDU output knowledge: E.g., why/how eco-disturbances happened (over-farming, deforestation, biomass burn)?
Output Information
fPar anomalies
SST clusters
Associations (e.g. SST~NINO, SOI~ NPP, fPAR~ NPP)
Output Knowledge
Significant ecosystem disturbance occurred. Disturbances are cause by climate changes
Catalog/Metadata
Algorithm/service catalog
Algorithm/service descriptions, input/output types, preconditions, consequences, etc.
D/I/KArchive store catalog
Type, content, potential use, lineage, format etc.
Archive store

Figure 4.6 – Detailed KBS/IA Architecture showing **KUMAR** flows

4.2 Key Components and Interfaces

4.2.1 Client/user interface layer

The first box at the top of Figure 4.5 is the client of the KBS/IA system. The client serves as the interface between user and the system. The client can be implemented either in web browsers or in any other graphical user interface. Users of the KBS/IA system perform searches for and access to various D/I/K products and services through the client. The major components of the client include the followings:

1) Interactive archive storage and algorithm/service discovery: the client interacts, through system and work flow control component, with the algorithm and archive storage catalogs in the system to search and locate available IDU algorithms, general purpose services, and D/I/K products, including virtual products. The algorithms, services and products can either be in one IA or in a number of different member IAs in case of a distributed or federated IA system.
2) interactive algorithm/service request, including specifications of parameters to the requested algorithm/service: upon the desired service being found through the

algorithm/service catalog, the client can request the service. If the requested service includes interactive parameters, the client will provide appropriate interfaces to the user to specify these parameters.

3) interactive stored and virtual product request: upon the desired stored D/I/K products or virtual products being found through the catalogs, the client can request the products. If a requested product is a virtual product, the related service or service chain will be invoked to materialize the virtual product.
4) interactive workflow editing: if the product the client requests is neither available in the stored D/I/K catalog nor in virtual product catalog, the user may interactively edit a workflow for the desired product. The client provides a graphical interface for the user to perform such editing.

In addition to the above four major components, other functionalities, such as data manipulation and project management, can also be added to the client but these "value-added" components should not be considered as inherent to the KBS/IA system.

In order for the client to interact with the KBS/IA's services, a common service environment, which is a set of standard interfaces within the KBS/IA system, is required. Depending on the scope of the KBS/IA, the standard interfaces can range from discipline- or community-specific to open web services oriented.

4.2.2 IDU algorithm and service layer

The algorithms and services in this layer can be grouped into four categories/components: 1) a system and workflow management component which performs system status monitoring, plans, constructs and manages workflows to produce new D/I/K products; 2) an IDU algorithm component which provides various IDU algorithms to stored and ingesting data; 3) a science model component which executing other scientific models such as atmospheric and hydrologic models; and 4) a general data service component which provides various general purpose data manipulation services such as subsetting, map reprojection, and format conversion. Among the four components, only the first two are necessary for a KBS/IA system while the last two are more general purpose services and thus may or may not exist in a KBS/IA system.

4.2.2.1 IDU algorithms

The IDU algorithm is the core component of the KBS/IA system, although other components, such as catalog and workflow control, are also necessary. Each IDU algorithm is implemented as one or more services (or agents, modules) that can be chained together to perform a specific IDU task. Ontology descriptions, such as input/output types, precondition, and consequences, are provided in service profile for each service/algorithm. Ontology descriptions also include knowledge, inference engine, relationships, etc. Depending on the nature of the IDU algorithms, there may or may not be interfaces between algorithms/services and the archive users. For example, interfaces are needed for certain supervised classifications in which interactive parameterizations are required, while no interface is required for some unsupervised clustering algorithm.

Algorithm interfaces are useful for experienced users who have domain knowledge and are familiar with the algorithms. For general users of the archive, who are interested in finding and obtaining D/I/K but are not specifically interested in how the D/I/K are derived, the algorithm interfaces are less important.

4.2.2.2 System and workflow control

The system and workflow control service component contains various services for monitoring, configuring, managing, and tasking software/hardware/data resources and for planning, constructing, managing and executing workflows. In most case, workflow construction is an automated process in which services are chained based on the user requirement and service descriptions. For example, a simple event detection algorithm of vegetation dynamics may use archived time series of a vegetation index as baseline to detect sudden increase/decrease in current vegetation activities. The workflow of this service can be automatically constructed based on the ontology descriptions of the vegetation index computation equation (e.g., ratio of near infrared reflectance to red reflectance), the index baseline computation equation (e.g., precedent 12-month values, precedent three-year averages), and the archived time series data. When new data is ingested into the archive system, the algorithm calculates the vegetation index and compares it with the baseline index value (which itself is time-variant as new data is ingested). If the difference is larger than a predefined threshold, a high/low anomaly alarm is then issued. The result of such event detection can in turn activate next step in the KBS/IA system, such as sensor re-tasking, especially in emergency management scenario (e.g., wild fire). While automatic workflow construction and execution is important in a KBS/IA system, an experienced user of the algorithm may want to manually construct the workflow and submit it to the system. Thus, interfaces between individual IDU algorithms and users are also required.

4.2.2.3 Other algorithms and science models

This component contains algorithms and science models that are not belong to the IDU algorithms. These include, for example, algorithm of calculating fraction of photosynthetically active radiation absorbed by vegetation (fPAR) and models of deriving net primary production, potential soil erosion, and wild fire risk index. These algorithms and models are not considered as necessary components of a KBS/IA system but they are associated, either directly or indirectly, with the IDU algorithms. On one hand, the output of these algorithms/models (e.g., fPAR) can serve as input of IDU algorithms. On the other hand, the output of IDU algorithms (e.g., data quality, clustering, fractal dimensions) can also be used as input as these algorithms/models.

4.2.2.4 General data service

The general data service component contains services that may be required for general data processing and are not specific to any particular IDU algorithm. These may include such services as spatial, temporal, and parameter subsetting, georectification and reprojection, data format conversion, and so on. Although this is not a necessary component of a KBS/IA system, the services provided in this component is often indispensable for an archive system.

4.2.3 Catalog

The catalog together with the system and workflow control system composes a special kind of service for a KBS/IA system, i.e., the registry service. This service provides necessary information for the KBS/IA clients to search, find, access, and obtain algorithms and services available from the system. It also enables algorithms/services implemented in the system to announce and publish themselves. The catalog provides three types of information: algorithm/service catalog, stored I/D/K product catalog, and virtual I/D/K product catalog. The algorithm/service catalog contains information about available services in the KBS/IA system, includes IDU and non-IDU algorithms, science models, and other general purpose data services. The stored I/D/K product catalog contains information about the products actually stored in the archive storage. These products are readily available to users without needing further processing. The virtual product catalog contains information about products that can readily be produced but are not materialized and stored in the archive storage. The virtual products are actually workflows that can, upon being executed, produce real products.

4.2.4 Archive Storage

This archive storage includes D/I/K which is kept in the storage area of an archive. The D/I/K included in this area can be accessed to by clients, algorithms/services, and catalog of the KBS/IA. For any specific IDU algorithm, the content in the archive storage can be distinguished as an input component and an output component. The input to an IDU algorithm is usually data or information and the output is usually information or knowledge. For different algorithms, the output of one can be the input of another. For example, the input to a spatial correlation data mining algorithm may be the output of an image segmentation algorithm or data clustering algorithm. The two arrows in the "archive D/I/K store" in figure 4.5 are examples of this input/output relationship. The content in the archive storage is accessed to by the catalog and various algorithms/services (see the two block arrows linking the storage box and the algorithm and the catalog boxes). A set of standard interface protocols are required to perform these accesses. Similar to the common service environment discussed in section 4.2.1, the scope of "standardization" of storage access can range from discipline- or community-specific to a wide open environment such as the entire internet.

4.3 Service and Data Flows

The service and data flows of IDU algorithms in a KBS/IA system is shown in figure 4.6, by taking Vipin Kumar's algorithms as examples. In this diagram, the service flows are indicated using green arrows. The input data flows are indicated using red arrows. The output data flows are indicated using the pink arrows. The linkage between the IDU algorithms and the algorithm knowledge base are shown in blue arrows. The dotted orange arrows indicate how the output of one IDU algorithm becomes the input of another IDU algorithm. The block arrows among different components are still kept in this diagram to show the interfaces among these components.

The main event/data flows include the followings:

1) A KBS/IA system user requests Kumar's algorithms through the system's client interface.
2) The client submits the user's information to system control service to conduct user authentication and authorization.
3) The workflow planner communicates with the algorithm catalog in the system catalog to find the specific IDU algorithm and the service profile of the algorithm, including precondition, input/output, and consequences of executing the algorithm.
4) The workflow planner communicates with the storage catalog in the system catalog to check the status of required input contents.
5) The workflow planner checks with the system status monitoring service and system resources managements services to learn the system status and resources availability in order to perform workflow construction.
6) The workflow construction service constructs an executable workflow required to complete the requested algorithm. The workflow construction service can also interact with the client to allow the requesting user to edit and construct the workflow.
7) The workflow execution service talks with system status and resources management to execute the workflow at optimum system status.
8) The IDU algorithm is invoked and executed.
9) The necessary input D/I/K to the algorithm is obtained from the archive storage or from the system ingesting data.
10) The output I/K of the algorithm is stored to the archive storage and/or returned to the request users through the client interface.
11) The catalog is updated with the latest addition to the archive store.
12) The outcome of the algorithm may also result in requests to observation system such as the sensor re-tasking due to the detection of a severe ecosystem disturbance (e.g., wild fire). This, in turn, may result in actions in the low level data processing system, the ingestion of new data into the archive store, and the update of the archive store catalog.

4.4 OWL-S Examples for Kumar's Algorithms

The Kumar's algorithms shown in figure 4.6 can be described using Web Ontology Language for Service (OWL-S) ([21]; [20]). The followings three OWL-S examples used to describe the algorithms: 1) top level service definition, which includes the references to service profile, service process model, and service process grounding; 2) service profile, which includes information of the service provider, service function description, service category, and service parameters; and 3) service process, which describes a service process and control model such as inputs, outputs, preconditions, effects, and component sub-processes.

4.4.1 Top level service definition

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE uridef (View Source for full doctype...)>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
           xmlns:owl="http://www.w3.org/2002/07/owl#"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
           xmlns:service="http://www.daml.org/services/owl-s/1.0/Service.owl#"
           xmlns="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Service.owl#">
    <owl:Ontology rdf:about="">
      <owl:versionInfo>$Id: Kumar__Pattern_Service.owl,v 1.0 2004/10/02 02:10:14 martin Exp $</owl:versionInfo>
      <rdfs:comment>This ontology represents the Kumar's pattern service description for the Intellignet Archive
example.</rdfs:comment>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Service.owl" />
      <owl:imports rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Profile.owl" />
      <owl:imports rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl" />
    </owl:Ontology>
    <service:Service rdf:ID="Kumar_Pattern_Service">
     <!--  Reference to the Profile -->
     <service:presents                                   rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Profile.owl#Profile_Kumar_Pattern" />
     <!--  Reference to the Process Model -->
     <service:describedBy                                rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Process.owl#Kumar_Pattern_ProcessModel" />
     <!--  Reference to the Grounding -->
     <service:supports                                   rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Grounding.owl#Grounding_Kumar_Pattern" />
    </service:Service>
  </rdf:RDF>
```

4.4.2 Service profile

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE uridef (View Source for full doctype...)>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
           xmlns:owl="http://www.w3.org/2002/07/owl#"
           xmlns:service="http://www.daml.org/services/owl-s/1.0/Service.owl#"
           xmlns:process="http://www.daml.org/services/owl-s/1.0/Process.owl#"
           xmlns:profile="http://www.daml.org/services/owl-s/1.0/Profile.owl#"
           xmlns:actor="http://www.daml.org/services/owl-s/1.0/ActorDefault.owl#"
           xmlns:addParam="http://www.daml.org/services/owl-s/1.0/ProfileAdditionalParameters.owl#"
           xmlns:profileHierarchy="http://www.daml.org/services/owl-s/1.0/ProfileHierarchy.owl#"
           xmlns="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Profile.owl#">
    <owl:Ontology rdf:about="">
      <owl:versionInfo>$Id: Kumar_Profile.owl,v 1.0 2004/10/3 02:10:14 martin Exp $</owl:versionInfo>
      <rdfs:comment>Kumar's Association Example for OWL-S Profile description</rdfs:comment>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Service.owl" />
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Profile.owl" />
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/ActorDefault.owl" />
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/ProfileAdditionalParameters.owl" />
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Process.owl" />
      <owl:imports rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Service.owl" />
      <owl:imports rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl" />
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/ProfileHierarchy.owl" />
    </owl:Ontology>
    <!--  ################################################################ -->
    <!--  # Instance Definition of Kumar's pattern service  #-->
    <!--  ################################################################ -->
    <profileHierarchy:Kumar_Pattern rdf:ID="Profile_Kumar_Pattern">
      <!--  reference to the service specification -->
```

```
   <service:presentedBy                              rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Service.owl#Kumar_Pattern_Service" />
   <!--  reference to the process model specification -->
   <profile:has_process                              rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Process.owl#Kumar_Pattern_ProcessModel" />
   <profile:serviceName>Kumar_Pattern_Service</profile:serviceName>
   <profile:textDescription>This service is to find spatio-temporal patterns in Earth science data.
     This typically involves associations, clusters, predictive models and trends</profile:textDescription>

   <profile:contactInformation>
     <actor:Actor rdf:ID="Kumar_Pattern">
       <actor:name>Lab for Advanced Information Technology and Standards</actor:name>
       <actor:title>Operator</actor:title>
       <actor:phone>301 552 3829</actor:phone>
       <actor:fax>301 552 9671</actor:fax>
       <actor:email>wyang1@gmu.edu</actor:email>
       <actor:physicalAddress>9801 Greenbelt Rd. Suit 316-317, Lanham, MD, 20706</actor:physicalAddress>
       <actor:webURL>http://ws.laits.gmu.edu/services/Kumar_Pattern.html</actor:webURL>
     </actor:Actor>
   </profile:contactInformation>
   <profile:contactInformation>
     <actor:Actor rdf:ID="IDU-information">
       <actor:name>Wenli Yang</actor:name>
       <actor:title>Principal Scientist</actor:title>
       <actor:phone>301 552 3829</actor:phone>
       <actor:fax>301 552 9671</actor:fax>
       <actor:email>wyang1@gmu.edu</actor:email>
       <actor:physicalAddress>9801 Greenbelt Rd. Suit 316-317, Lanham, MD, 20706</actor:physicalAddress>
       <actor:webURL>http://ws.laits.gmu.edu/services/Kumar_Pattern.html</actor:webURL>
     </actor:Actor>
   </profile:contactInformation>

   <!--  Specification of the service category using NAICS -->
   <profile:serviceCategory>
     <addParam:ISO19119 rdf:ID="ISO19119-category">
       <profile:value>Data Mining</profile:value>
       <profile:code>****</profile:code>
     </addParam:ISO19119>
   </profile:serviceCategory>

   <!--  Descriptions of IOPEs -->
   <profile:hasInput rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl#fPAR" />
   <profile:hasInput rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl#SST" />
   <profile:hasInput rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl#NPP" />
   <profile:hasInput                                 rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Process.owl#Climate_Indices" />
   <profile:hasInput                                 rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Process.owl#Land_Cover" />
   <profile:hasInput                                 rdf:resource="http://ws.laits.gmu.edu/services/owl-
s/1.0/Kumar/Kumar_Process.owl#Association_Rules" />
  </profileHierarchy:Kumar_Pattern>
 </rdf:RDF>
```

4.4.3 Service process

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[ ]>


<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:service="http://www.daml.org/services/owl-s/1.0/Service.owl#"
     xmlns:process="http://www.daml.org/services/owl-s/1.0/Process.owl#"
     xmlns:profile="http://www.daml.org/services/owl-s/1.0/Profile.owl#"
```

```
        xmlns:actor="http://www.daml.org/services/owl-s/1.0/ActorDefault.owl#"
        xmlns:addParam="http://www.daml.org/services/owl-s/1.0/ProfileAdditionalParameters.owl#"
        xmlns:profileHierarchy="http://www.daml.org/services/owl-s/1.0/ProfileHierarchy.owl#"
        xmlns:concepts = "http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/concepts.owl#"
        xmlns="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Process.owl#">

 <owl:Ontology rdf:about="">
   <owl:versionInfo>$Id: Kumar_Process.owl,v 1.0 2004/10/03 23:06:47 martin Exp $</owl:versionInfo>
   <rdfs:comment> Kumar's pattern Example for OWL-S Process Model</rdfs:comment>
   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Service.owl" />
   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Profile.owl" />
   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Process.owl" />
   <owl:imports rdf:resource="http://ws.laits.gmu.edu/services/owl-s/1.0/Kumar/Kumar_Service.owl" />
 </owl:Ontology>

 <!-- ################################################################ -->
 <!-- Instance Definition of Kumar's Pattern Process Model -->
 <process:ProcessModel rdf:ID="Kumar_Pattern_ProcessModel">
   <process:hasProcess rdf:resource="#Kumar_Process" />
   <service:describes rdf:resource="#Kumar_Pattern_Service"/>
 </process:ProcessModel>


 <!-- ###################################################################### -->
 <!-- Definition of top level Process as a composite process -->


<process:CompositeProcess rdf:ID="Kumar_Process">
   <rdfs:label> This is the top level process for Kumar's pattern </rdfs:label>
   <process:composedOf>
      <process:Sequence>
        <process:components rdf:parseType="Collection">
          <process:AtomicProcess rdf:about="#Ecosystem_Disturbance>
        <process:AtomicProcess rdf:about="#SNN_Clustering"/>
        <process:CompositeProcess rdf:about="#Correlation_Association/>
      </process:components>
     </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>


 <!-- ###################################################################### -->
 <!-- Ecosystem disturbance (ATOMIC)-->
 <process:AtomicProcess rdf:ID="Ecosystem_Disturbance">
    <process:hasInput rdf:resource="#fPAR"/>
    <process:hasOutput rdf:resource="#fPAR_Anomalies_Out"/>
 </process:AtomicProcess>
 <process:Input rdf:ID="fPAR">
    <process:parameterType rdf:resource="&concepts;#fPAR"/>
 </process:Input>
 <process:UnConditionalOutput rdf:ID="fPAR_Anomalies_Out">
   <process:parameterType rdf:resource="&concepts;#fPAR_Anomalies_Out"/>
 </process:UnConditionalOutput>

 <!-- ###################################################################### -->
 <!-- SNN Clustering (ATOMIC)-->
 <process:AtomicProcess rdf:ID="SNN_Clustering">
    <process:hasInput rdf:resource="#SST"/>
    <process:hasInput rdf:resource="#NPP"/>
    <process:hasInput rdf:resource="#Climate_Indices"/>
    <process:hasInput rdf:resource="#Land_Cover"/>
    <process:hasOutput rdf:resource="#SNN_Clustering_Out"/>
 </process:AtomicProcess>
 <process:Input rdf:ID="SST">
   <process:parameterType rdf:resource="&concepts;#SST"/>
 </process:Input>
 <process:Input rdf:ID="NPP">
  <process:parameterType rdf:resource="&concepts;#NPP"/>
 </process:Input>
 <process:Input rdf:ID="Climate_Indices">
  <process:parameterType rdf:resource="&concepts;#Climate_Indices"/>
 </process:Input>
 <process:Input rdf:ID="Land_Cover">
  <process:parameterType rdf:resource="&concepts;#Land_Cover"/>
 </process:Input>
 <process:UnConditionalOutput rdf:ID="SST_Clustering_Out">
  <process:parameterType rdf:resource="&concepts;#SST_Clustering_Out"/>
 </process:UnConditionalOutput>

 <!-- ############################################################################## -->
 <!-- Correlation & Association (ATOMIC) -->
 <process:AtomicProcess rdf:ID="Correlation_Association">
  <process:hasInput rdf:resource="#fPAR_Anomalies_Out"/>
  <process:hasInput rdf:resource="#SST_Clustering_Out"/>
  <process:hasInput rdf:resource="#Association_Rules"/>
  <process:hasOutput rdf:resource="#Association_Out"/>
 </process:AtomicProcess>
 <process:Input rdf:ID="Association_Rules">
  <process:parameterType rdf:resource="&concepts;#Association_Rules"/>
 </process:Input>
 <process:UnConditionalOutput rdf:ID="Association_Out">
  <process:parameterType rdf:resource="&concepts;#Association_Out"/>
 </process:UnConditionalOutput>
</rdf:RDF>
```

5. Conclusion

The work described in this paper represents both the capstone of work that has been ongoing for some time as well as a transition to the next phase of the research: prototyping and demonstration. As a capstone document, it clearly shows how the IS/IDU research and development projects support the key functional requirements for an intelligent archive as part of a knowledge building system. The paper also shows that the necessary critical technology infrastructure is in place; and that reasonable achievable software implementations can be built to support and use the algorithms, on the one hand, and to fully utilize the capabilities provided by existing infrastructure, on the other. The Use Case discussions wrap this together in a compelling way, showing how a variety of powerful and very useful capabilities and concepts of operation can be supported by the envisioned IA conceptual specimen architecture.

With this as a basis, the next step appears to be to show that this vision can be realized in practice. By prototyping and demonstrating selected challenging sub-functions using IS/IDU algorithms on real data, it will be possible to make a strong case that the next generation of intelligent archive technology is achievable, and will provide quantum levels of improvement over current capabilities at acceptable cost and risk.

References:

Papers Relating to Intelligent Archives

All the following papers are available at:
http://daac.gsfc.nasa.gov/IDA/presentations.shtml

[1] "Conceptual Study Of Intelligent Archives of the Future," Ramapriyan, X. et al., NASA Technical Report, August 2003

[2] "Intelligent Archive Visionary Use Case: Advanced Weather Forecast Scenario," Harberts, R. and Roelofs, S., NASA Technical Report, July 2003

[3] "Intelligent Archive Visionary Use Case: Precision Architecture Scenario," Harberts, R. and Roelofs, S., NASA Technical Report, July 2003

[4] "Intelligent Archive Visionary Use Case: Virtual Observatories," Harberts, R. and Roelofs, S., NASA Technical Report, July 2003

[5] "Moving from Data and Information Systems to Knowledge Building Systems: Issues of Scale and Other Research Challenges," McConaughy, G. and McDonald, K, NASA Technical Report, September 2003

[6] "Optimizing Performance in Intelligent Archives," Morse, H. and Isaacs, D., NASA Technical Report, January 2003

[7] "Virtual Data Products in an Intelligent Archive," Clausen, M. and Lynnes, NASA Technical Report, July 2003.

[8] "Automated Data Quality Assessment in the Intelligent Archive," Isaac, D. and Lynnes, C., NASA Technical Report, January 2003.

[9] "Assessment of the Applicability of IDU Research and Technologies to a Proposed Intelligent Archive Test Bed," Morse, S., NASA Technical Report, July 2004

Papers relating to Cyberinfrastructure and IA Software Architecture

[10] Foster I., C. Kesselman, J.M. Nick and S. Tuecke, 2002. The Physiology of the Grid: An open Grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum.
http://www.globus.org/research/papers/ogsa.pdf.

[11] Foster I., C. Kesselman and S. Tuecke, 2001. The Anatomy of the Grid – Enabling Scalable Virtual Organizations. Intl. J. of High Performance Computing Applications, 15(3), 200-222.

[12] Foster, I. and C. Kesselman, editors, 1999. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers.

[13] Foster I. and C. Kesselman 1998. The Globus Project: A Status Report. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18. ftp://ftp.globus.org/pub/globus/papers/globus-hcw98.pdf

[14] GGF, 2004. The WS-Resource Framework. http://www.globus.org/wsrf/.

[15] Globus, 2004. Globus homepage, http://www.globus.org

[16] Kumar, V., Steinbach, M., Tan, P., Potter, C., and Klooster, S., 2004, Discovery of Changes from the Global Carbon Cycle and Climate System Using Data Mining, http://www.esto.nasa.gov/conferebces/estc2004/papers/b9p2.pdf

[17] Levent, E., Steinbach, M., and Kumar, V., 2003, Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data, SIAM International Conference on Data Mining, http://www-users.cs.umn.edu/~kumar/papers/SIAM_snn.pdf

[18] OASIS, 2004. Organization for the Advancement of Structured Information Standards (OASIS) homepage. http://www.oasis-open.org/home/index.php.

[19] Tan, P., Steinbach, M., Kumar, V., Potter, C., Klooster, S., and Torregrosa., A., 2001, Finding Spatio-Temporal Patterns in Earth Science Data, KDD 2001 Workshop on Temoral Data Mining, http://www-users.cs.umn.edu/~kumar/papers/spatio_temoral15.pdf

[20] The DAML Services Coalition, 2002, DAML-S: Semeantic Markup for Web Services, http://www.daml.org/services/daml-s/0.7/daml-s.html

[21] The OWL Services Coalition, 2003, OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.0/owl-s.pdf

[22] W3C, 2004. World-Wide Web Consortium (W3C) homepage, http://www.w3c.org

[23] Zhang, P., Huang, Y., Shekhar, S., and Kumar, V., 2003, Correlation Analysis of Spatial Time Series Databases: A Filter-and-Refine Approach, Proceedings of the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining, Seoul, Korea, http://www-users.cs.umn.edu/~kumar/papers/pakdd03a.pdf